Achieving Global Coherence By Exploiting Conflict: A Distributed Framework for Job Shop Scheduling

PhD Thesis

Carla Pedro Gomes

Department of Artificial Intelligence
Faculty of Science and Engineering
University of Edinburgh

1992

Abstract

The work presented in this thesis is motivated by the "hard" nature of the job shop scheduling task due to the "intrinsic" and "extrinsic" complexity of realistic problems. A scheduling framework, combining Artificial Intelligence and Operations Research techniques and implemented in a Distributed Problem Solver environment suitable for parallel implementation, is described. The adopted approach views the system as an *Organisation.* Agents are assigned different roles and functions depending on their position within the structure of the *Organisation.* In this Organisation, agents of the same level state their interests independently of each other and therefore *Conflict* is likely to occur. A major thesis of the research reported here is that not only is it important to deal with *conflict* but also that *conflict* as a consequence of the scheduling process should be exploited as a way of integrating different scheduling perspectives, as a way of allowing agents to express their own interests independently of each other and, thus, guaranteeing *pluralism. Pluralism* is also ensured by providing agents with both empirical knowledge (heuristics, dispatch rules) and theoretical knowledge (optimal algorithms) and by explicitly allowing the coexistence of a *job based perspective*, a *resource based perspective* and an *operation based perspective* enabling so called *opportunistic* and *micro-opportunistic* scheduling. In order to achieve *Global Coherence* in this conflicting distributed environment, agents are provided with mechanisms to make them aware of the structural and intrinsic features of the (sub)problems that they have to solve and the interaction of their (sub)problems, without relying on communication with each other, and with tools to analyse, evaluate and solve the *conflicts. Structural Awareness* is a major concept introduced and developed in the research reported in this thesis.

Acknowledgements

First I thank my supervisors, Austin Tate and Lyn Thomas, for all their work on my behalf. They have listened to, read and criticised my ideas from an embryonic stage to their implementation. Tim Smithers was also my supervisor for a short while. I thank him for that.

I also thank Roberto Desimone and Ken Currie for the highly useful conversations, ideas and advice during the first year of this research.

Several people have made comments on drafts, given constructive criticism, helped with programming issues or proof-read infinite versions of this thesis. Perhaps I might mention in particular Alan Black, Richard Caley, Jeff Dalton, Ian Lewin, Nelson Ludlow, Gary Roberts, Sarah Price and Wamberto Vasconcelos.

Thanks to the "E17 gang and associates", Alan Black, Richard Caley, Rolando Carrera, Matt Crocker, Lynn Jamieson, Alvaro Fernandes, Regina Fernandes, Ian Frank, Khee-Yin How, Alistair Knott, Ian Lewin, Nelson Ludlow, Suresh Mananhdar, Dave Moffat, Keiichi Nakata, Sarah Price, Brian Ross, Gary Roberts, Flávio Corrêa da Silva, Rob Scott and Wamberto Vasconcelos, for their friendship, companionship and for the academic milieu, not to mention their understanding for my insatiable eating habits.

Nelson Ludlow deserves a special mention. He was *the* incentive for me to finish this work *asap*.

Maria Alice and Octávio engendered and encouraged my curiosity for finding out what a PhD is about. They also gave me their unconditional support while I was doing this research. They always believed that their daughter would get a PhD some day. I hope not to disappoint them!

Professor Gomes Cardoso has always encouraged me in my academic career and, above all, he has always been a friend.

Financial support for this work was provided by INDEG-ISCTE, Portugal, for the first 3 months of this research, and by Junta Nacional de Investigação Científica, Portugal, in the form of a grant, for the years of 1990, 1991 and 1992.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Carla Pedro Gomes
Edinburgh
October 27, 1992

Contents

List of Figures

Chapter 1

Introduction

A distributed problem solver combining Operations Research and Artificial Intelligence techniques is described in this thesis. This framework is suitable for a *Job Shop Scheduler*, in particular to intermix multiple scheduling perspectives, to integrate predictive capabilities with reactive capabilities and to combine optimal models to solve the mathematically well defined components of the job shop scheduling problem with more heuristic type models to tackle the more qualitative components of the problem. EXPLICIT is the job shop scheduling framework that embodies the research described in this thesis.

The approach adopted in EXPLICIT views the system as an *Organisation.* Agents are assigned different roles and functions depending on their position within the structure of the *Organisation.* Top level agents are responsible for global and structural functions (and consequently decisions), while more detailed and specific tasks and functions are assigned to the bottom level agents. In this *Organisation*, agents of the same level state their interests independently of each other.

EXPLICIT can be compared to a hierarchical organisation with three main levels: the Strategic level, the Tactical level and the Operational level. This structure is inspired by DAS (Distributed Asynchronous System), a system developed at the University of Strathclyde [Buchanan *et al* 88], [Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89]. However, although there are some similarities between EXPLICIT and DAS in terms of the general structure of the system, there are substantial differences in

terms of the processes associated with the different agents of the systems, i.e., the functional organisation of the systems, and in terms of the techniques and methods used in both systems. In EXPLICIT, the functions of *analysis, planning* and *evaluation* are reinforced and explicitly assigned to the different agents of the system.

[Kornfeld & Hewitt 88] introduced the idea of the "scientific community metaphor". The success of scientific research depends heavily on complementary perspectives and knowledge, in order to allow *pluralism.* In EXPLICIT, ensuring *pluralism* corresponds to integrating different scheduling perspectives. In EXPLICIT, *pluralism* is ensured by allowing agents to perform their scheduling decisions independently of each other, by providing agents with both empirical knowledge (heuristics, dispatch rules) and theoretical knowledge (optimal algorithms) and by explicitly allowing the coexistence of a *job based perspective*, as in ISIS [Fox & Smith 84], a *resource based perspective*, as in OPT [Lawrence 87] or RESS-II [Liu 88], and an *operation based perspective*, as in MICRO-BOSS [Sadeh 91] or in PCP [Berry 91], enabling so called *opportunistic* scheduling, as in OPIS [Smith 87], [Smith *et al* 86] , and *micro-opportunistic* scheduling, as in MICRO-BOSS.

Conflict is a central concept in EXPLICIT. The approach adopted is to consider *conflict* as a normal and even "healthy" state within a system, as long as it is "under control". *Conflict* arises as a consequence of the dynamic nature of the environment. Furthermore, *conflict* can arise as a consequence of the scheduling process. The first form of *conflict* is "non-negotiable" and it is imposed by the environment. Managing this type of *conflict* corresponds to providing the system with reactive capabilities. The latter form of *conflict* arises from the solution process. A major thesis of the research described here is that not only is it important to deal with *conflict* but also that *conflict* as a consequence of the scheduling process should be exploited as a way of integrating different scheduling perspectives. In EXPLICIT, *conflict* as a consequence of the scheduling process is stimulated as a form of guaranteeing *pluralism*, i.e., a way of allowing the agents of the system to perform their best scheduling decisions considering their own interests. In this sense, exploiting *conflict* corresponds to allowing each agent to express its own interests independently of the others. Detecting and solving

conflict corresponds to coordinating and making compatible the conflicting interests of the agents.

Considering the *conflicting* nature of the scheduling environment, agents are provided with several mechanisms to make them *structurally aware* of the (sub)problems that they have to solve and the interaction of their (sub)problems, without relying on communication with each other, and with tools to analyse, evaluate and solve the *conflicts.* Considering the distributed problem solving point of view, the research described in this thesis investigates strategies for ensuring *Global Coherence* and *Conflict Resolution* within a distributed environment.

EXPLICIT integrates optimal models for solving the mathematically well defined components of the (sub)problems its agents have to solve with more heuristic type procedures for the qualitative components of the system.

EXPLICIT combines a conflict resolution algorithm, a critical path method (see, e.g., [Willis 85], [Davis & Patterson 75] and [Fendley 68]), an assignment based algorithm, which integrates an assignment algorithm (see, e.g., [Gondran & Minoux 84]) and an optimal algorithm for minimising maximum lateness ([McMahon & Florian 75]). EXPLICIT performs a sequence of optimisations of the load balancing of resources, each time assigning times and resources to the operations considered more critical. Each operation is analysed individually (*operation based perspective*) and its criticality results from the combination of a *job based perspective* with a *resource based perspective.*

A cases performance analysis comparing the results produced by EXPLICIT with the results produced by four popular dispatch rules is described in this thesis. The results of the comparison of EXPLICIT with the four popular dispatch rules are very encouraging. EXPLICIT clearly outperforms the four dispatch rules, especially in terms of tardiness and utilisation of resources, the two main objectives embodied in the system.

Plan of The Thesis

- Chapter 2 - The job shop scheduling problem is defined in this chapter. The complexity of the scheduling problem is categorised into two perspectives: the

intrinsic and the *extrinsic* point of view. A review of the state of the art of the job scheduling problem and related work is also presented in this chapter. Weak points and potential areas for research are highlighted.

- Chapter 3 - The major targets of the research described in this thesis are introduced in this chapter: the integration of multiple scheduling perspectives; the integration of predictive capabilities with reactive capabilities; increasing optimisation rather than settling for satisfaction in a scheduling framework. Some fundamental issues underlying the approach adopted are discussed: the assignment of different roles and skills to agents within a distributed problem solver framework; ensuring *pluralism*; the merge of Operations Research and Artificial Intelligence techniques ; ensuring *structural awareness*; and providing agents with mechanisms to detect and solve *Conflict.* Considering a distributed problem solving point of view, the research described in this thesis investigates strategies for ensuring *Global Coherence* and *Conflict Resolution* within a distributed problem solving environment.

- Chapter 4 - EXPLICIT is the name of the job shop scheduling framework described in this thesis. EXPLICIT stands for "exploiting conflict". The overall structure of EXPLICIT is presented in this chapter. The scheduling process is outlined. Two central concepts embodied in EXPLICIT are also introduced: the concept of *conflict* and the concept of *structural awareness.*

- Chapter 5 - A detailed description of the scheduling process adopted in EXPLICIT is presented in this chapter. The entities and scheduling functions assigned to the agents of the system are described in detail. Representation issues and algorithms are given. The concept of conflict, which is central to EXPLICIT, is formally defined. The characteristics of the types of problems that can be solved within the proposed framework are also discussed.

- Chapter 6 - EXPLICIT is analysed from an organisational point of view, i.e., in terms of roles and organisational functions assigned to the different agents of the system, its communication and information flows and its control regime.

- Chapter 7 - A simple example is comprehensively described in this chapter. The main purpose of the example is to illustrate the scheduling process adopted in EXPLICIT. The scheduling process is described focusing on the roles performed by the different agents of the system and their interaction. Particular emphasis is given to the process of conflict detection and conflict resolution adopted within EXPLICIT.

- Chapter 8 - A comparison of the approach adopted in EXPLICIT with other approaches is presented in this chapter. Particular emphasis is given to the comparison between EXPLICIT and DAS, since EXPLICIT is mainly inspired by DAS's architecture.

- Chapter 9 - A case performance analysis is presented in this chapter. The performance analysis compares the results produced by two different versions of EXPLICIT with the results produced by four dispatch rules. EXPLICIT outperforms the four dispatch rules, especially in terms of tardiness and utilisation of resources, the two main objectives embodied in the system. The time performance of EXPLICIT is also analysed in this chapter. This chapter also includes the description of the generator of the battery of cases used for the performance analysis.

- Chapter 10 - This chapter summarises the main contributions of the research described in this thesis. A discussion of future possible research directions is also included in this chapter.

Chapter 2

The Job Shop Scheduling Problem - A Review

In this chapter the job shop scheduling problem is defined. The complexity of the scheduling problem is categorised into two perspectives: the *intrinsic* and the *extrinsic* point of view. A review of the state of the art of the job shop scheduling problem and related work are also presented in this chapter. Weak points and potential areas for research are highlighted.

2.1 The Job Shop Scheduling Problem

2.1.1 The Problem Definition

Scheduling is defined by [Baker 74] as:

> The allocation of resources over time to perform a collection of tasks.

This thesis concerns a particular type of scheduling, job shop scheduling. The job shop scheduling problem (JSSP) consists of assigning times and individual machines to a set of jobs that have to be performed on a finite set of resources. Each job, also called order, consists of a set of operations related to each other according to a certain process plan that specifies a partial ordering among the operations. The process plan of each job does not have to be the same.

Historically, the JSSP has been divided into two separate steps: the process of routing, also referred to as planning, and the process of assignment of times and resources to the operations or tasks, also referred to as scheduling. The process of planning consists of the selection of the sequence in which operations should be performed. This "historical" separation reflects the way the scientific community has tried to solve the problem in order to transform an "intractable" problem into two less "intractable" problems. [Fox & Smith 84] points out that this separation is somewhat fuzzier, as the selection of a process routing cannot be made conclusively without generating the accompanying schedule. The approach proposed in this thesis interleaves planning and scheduling.

2.1.2 Why is the JSSP a "Hard" Problem?

The problem that schedulers face can be stated as how to schedule and dispatch work in such a way that many stated and unstated conflicting goals are satisfied using information that is possibly incomplete, ambiguous, biased, outdated and erroneous. Additionally, the shop is seldom stable for longer than "half an hour".

The complexity of the JSSP can be categorised into two types:

- *Intrinsic* complexity
  - NP-Complete problem
- *Extrinsic* complexity
  - The vast number of constraints (hard and preferential)
  - Several goals usually interacting in conflicting and unforeseen ways (difficulty in evaluating the solutions)
  - Dynamic and stochastic environment

The *intrinsic* complexity refers to the complexity of the JSSP problem itself once formulated. The JSSP is NP-complete in the strong sense [Garey & Johnson 79]. The fascinating nature of NP-Complete problems represents a big challenge and, at the

same time, a pessimistic conjecture - one which predicts that polynomial time exact algorithms will never be found for this particular class of problems. The NP-Complete class is a subclass formed of the hardest problems in the class NP. For, if someone finds out a polynomial time exact algorithm to solve one of the problems belonging to this class (NP-Complete), then all the problems belonging to the class NP can be solved with a polynomial time exact algorithm. The "scientific feeling" is that this algorithm will never be found. However, this prognosis (P is different from NP) is only a conjecture and although pessimistic the research community should not give up from trying to find out the solution to tackle the NP-Complete problems and to show that P and NP are the same class. The practical consequence of dealing with a NP-complete problem is that job shop scheduling problems with realistic dimensions cannot be optimally solved in a "reasonable" amount of time, even assuming that a rigorous mathematical formulation of the problems can be found, considering the current state of the art.

The *extrinsic* complexity refers to the formulation of the JSSP. It relates to identifying the constraints and goals that affect the validity and quality of schedules. Hard constraints, if violated, invalidate a schedule. An example of a hard constraint is that two operations cannot be scheduled at the same time on a certain machine, if the machine can only process one operation at a time. While most hard constraints are obvious, they are also numerous and easy to overlook when manually planning and scheduling vast amounts of data. Preferential constraints are much more difficult to detect and incorporate in automatic systems. What appear to be minor details can have significant impact on the quality of the schedule. An error in the judgment can be much less obvious than a hard constraint violation and, while not invalidating the schedule, its quality might result very poor. The *extrinsic* complexity of the JSSP also refers to dealing with the dynamic and stochastic nature of the environment, which means that quite often plans and schedules have to be changed due to unforeseen events like breakdowns. Dealing with an uncertain environment requires a compromise between *predictiveness* and *reactiveness*. *Predictiveness* is the capability of making the best scheduling decisions assuming that unexpected events will not occur, while *reactiveness* is the capability of adapting an existing schedule to the constants changes that happen in the environment.

2.2 The State of the Art of the JSSP

An overview of the research in JSSP is presented in this chapter. Systems considered representative are succinctly described. The benefits and limitations of the current state of the art are highlighted. Chapter 8 presents a detailed comparative analysis of the framework proposed in this thesis with existing systems.

2.2.1 An Overview

The interest in scheduling problems in general and in the JSSP in particular is not recent. Different fields or research areas such as Operations Research, Information Theory, Control Theory and Artificial Intelligence have studied the JSSP from different perspectives.

During the 1950s, 1960s and 1970s the Operations Research (OR) community became interested in planning and scheduling problems in general and in the JSSP in particular. A lot of OR research has been done in the JSSP field (e.g., [Gere 66], [Kan 76], [Graves 81], [Blackstone *et al* 82], [French 82], [Lawler *et al* 82], [Dekel & Sahni 83], [Park *et al* 84], [Blazewicz88 *et al* 88], [McKay *et al* 88], [Buxey 89], [Cheng & Sin 90], [Blazwicz *et al* 91]). Until relatively recent times, the OR community concentrated their efforts on exact approaches which would guarantee finding optimal solutions. However, during the past two decades, several important results originated a change in attitude towards combinatorial problems, particularly the JSSP. In the early 1970s papers by [Cook 71] and [Karp 72] on NP-Completeness led to a significant change in attitudes. The effect of this work was for the OR community to turn from trying to devise exact algorithms for these problems to devising polynomial time heuristics algorithms, often with some sort of guarantees on the quality of the solution they produce with comparison to the optimum solution [Eglese 86].

Concomitantly, the Artificial Intelligence (AI) community became interested in combinatorial problems, providing new perspectives to their resolution. Particularly, combinatorial problems research was moved to real world environments with a special emphasis to capturing the human expertise to solve real world problems. The inter-

est from the AI community in the JSSP problem is relatively recent. The ISIS system [Fox & Smith 84] is considered the pioneer of the AI scheduling systems. However, since then, the interest in the JSSP by the AI community has increased substantially. Additionally, AI approaches provided a much more realistic dimension to the JSSP than the traditional optimal OR approaches.

Unfortunately, a satisfactory solution to real world JSSPs is far from being achieved. The current JSSP state of the art hasn't shown the decisive qualitative improvement required to tackle its complexity (*intrinsic* and *extrinsic*). Nevertheless, there seems to be a consensus that a multidisciplinary approach is essential to trying to tackle this problem. Although through different routes, both OR and AI community seem to agree in the importance and complementary role played by heuristic approaches and exact approaches. On one hand, heuristic approaches are used not only to tackle the NP-Complete nature of these problems (*intrinsic* complexity) but also to structure and understand their real world applications (*extrinsic* complexity)[1]. However, it should be noted that even with heuristic approaches there are no guarantees of avoiding computational complexity, if one wants to guarantee quality of the results. On the other hand, the research on exact algorithms leads to a better understanding of the problems and even to giving some insights to the development of new heuristics.

It is very common to classify scheduling systems according to the following framework:

- OR based systems
  - optimal systems
  - heuristic-based systems
- AI based systems

The first OR group includes systems that guarantee the optimal solution. Clearly, research in optimal solutions for the JSSP has a theoretical nature. Normally this type of research is applied to very specific situations, often single machine

[1]Actually, it is interesting to mention that the interest in heuristics started with [Polya 48] and [Simon & Newell 58] who predicted that "heuristic problem solving would be the next advance in operations research"[Simon & Newell 58].

problems (e.g., [Kan 76], [French 82], [Gupta & Kyparisis 87], [Dileepan & Sen 88], [Blazwicz *et al* 91]). Even for this simple case several (unrealistic) assumptions have to be adopted, such: no break downs, jobs are never interrupted, job operations never overlap, etc. In general this type of research leads to complicated formulae, most of them cannot be solved or they can only be solved for "small" problems. However, even with all these limitations, we cannot neglect this branch of research. On the contrary, much of the conceptual work in the JSSP area has its main source on this type of research. Additionally, this type of work provides insight into possible heuristic methods (e.g., [French 82], [Eglese 86]). Typical OR techniques used in the systems in this category include dynamic programming, branch and bound, mixed integer programming, cutting plane, restriction and relaxation techniques, construction and decomposition methods, etc (e.g., [Kan 76], [Graves 81], [French 82], [Dekel & Sahni 83], [Cheng & Sin 90], [Blazwicz *et al* 91]).

The second OR group includes systems that combine most of the techniques used by the first OR group and heuristic dispatching rules. Dispatching rules are used to provide rules to choose the next job, operation, resource, etc. Systems in this category do not guarantee an optimal solution but normally there is an attempt to "measure" the distance between the solution produced by the system and the optimal solution, for instance using worst case analysis. This area of OR is very rich, though normally the existing applications are tailored for particular problems. Nevertheless, this area constitutes an important source of techniques and models to tackle real applications of scheduling problems (e.g., [Gere 66], [Kan 76], [Graves 81], [Blackstone *et al* 82], [French 82], [Lawler *et al* 82], [Dekel & Sahni 83], [Park *et al* 84], [McKay *et al* 88], [Blazewicz88 *et al* 88], [Buxey 89], [Cheng & Sin 90], [Blazwicz *et al* 91]).

The third group consists of AI systems. The AI community's attitude towards the JSSP is completely different from the OR community's attitude. A major principle adopted in the design of AI scheduling systems is "satisfaction not optimality". The result is that AI scheduling systems tend to be more "knowledge based", more flexible, much more "general-purpose" and therefore much more adaptable to real world environments. However, the other side of the coin is that normally there is not much guarantee about

the quality and robustness of the solution and general performance of the system.

[Charalambous & Hindi 91] provides a good review of the research that has been done in JSSP in AI. A good review of the AI research that has been done in planning and scheduling (not only in the JSSP) can be found in e.g., [Allen *et al* 90] and [DARPA 90]. AI scheduling techniques to tackle the JSSP are still incipient, not supported by unifying theories. Therefore, it is difficult to identify clear AI approaches to the JSSP and to present systematic references. Many of the techniques used in AI scheduling systems are general problem-solving techniques and techniques from several AI (sub)areas. In particular, AI scheduling techniques include: constraint-based reasoning, constraint satisfaction formulation and constraint analysis, means end analysis, least commitment (e.g hierarchical search and non-linear planning, hierarchical decision making and priorities), interaction detection among goals, temporal coherence, typed preconditions, evaluation functions, temporal reasoning, belief revision and modal truth criteria, temporal coherence, question answering, opportunistic reasoning and other techniques such as beam search, one-then-best backtracking, dependency directed backtracking (see e.g., [Tate *et al* 90], [Desimone *et al* 90], [Drummond & Tate 89], [Charniak & McDermott 85])

2.2.2 Systems & Approaches to the JSSP

Due to the NP-Complete nature of the JSSP, efficient, exact and general purpose algorithms to solve this problem are nonexistent.

Traditionally, in real world environments, the JSSP has been solved using *priority dispatch rules* which consist of local rules to generate a schedule via a forward simulation of the shop (e.g., [Gere 66], [Panwalkar & Iskander 77], [Blackstone *et al* 82], [Park *et al* 84], [Kurtulus & Narula 85]). *Dispatch rules* generate a solution without any backtracking. A simple simulation model of the shop generates the scheduling events which correspond to the arrival of jobs to the shop and the completion of the processing of operations on the machines. Whenever a machine becomes free, a scheduling decision based on a dispatch rule is taken in terms of which operation to perform next.

Conventional systems, as opposed to AI systems since they rely heavily on OR heuristics, include MRP and OPT [Lawrence 87].

MRP stands for "Material Requirements Planning" and it was introduced in the 1960s. MRP is essentially a long term planning tool rather than a scheduling system. Basically, the MRP consists of the generation of the requirements in terms of end products for a certain period, the master production schedule, followed by the materials requirements planning (MRP). The materials requirements planning is a list of the required materials and parts to manufacture the planned production of end items. Parts already in stock are taken into consideration. The final step of the MRP approach consists of backward scheduling to calculate when the production of the required parts should commence [Hastings *et al* 82]. The main criticism of this approach is that it considers very unrealistic assumptions such as fixed lead times for each part, infinite capacity of the resources and no preventive maintenance.

OPT, "Optimised Production Technology" is by far the most popular scheduling system that was developed during the 1980s [Jacobs 84]. OPT tried to overcome the limitations of other production planning approaches.
OPT emphasises a *resource based perspective.* In a *resource based perspective*, the scheduling process is viewed from the point of view of the resources: there is a collection of resources on which operations should be scheduled. The choice of an operation to be assigned to a particular resource that becomes free is made from the set of eligible operations for that resource at the considered time. OPT distinguishes between bottleneck resources and non-bottleneck. Bottleneck resources should be used to determine the production rate of the whole factory - they "beat the drum". In OPT the scheduling process involves different steps. Initially backward scheduling is performed assuming infinite production capacity - bottleneck resources are detected at this phase. After this phase, and considering finite loading, a routine forwards schedules on the critical resources along with the work required after a job has been processed on the critical resource. The last step consists of backward scheduling the remaining job operations.

ISIS [Fox & Smith 84] is the pioneer of the AI scheduling systems, developed at CMU during the 1980s. ISIS represents a major change in terms of the JSSP since it uses a

framework much more adaptable to real world environments than previous scheduling systems. It models the shop through a very rich set of constraints in order to capture the several conflicting objectives and restrictions that occur within the factory (*intrinsic* complexity). The constraint representation formalism is used to support constraint hierarchies and constraint relaxation. The main contribution of the ISIS system is focused on: 1) constructing a knowledge representation that captures the job shop environment and its constraints to support constraint-directed-search and constraint relaxation; 2) developing a search architecture capable of exploiting this constraint knowledge to effectively control the combinatorics of the underlying search space.

ISIS creates schedules in an incremental manner and uses a *job based perspective.* A *job based perspective* views the scheduling problem as a collection of jobs that have to be scheduled. Once a job is selected to be scheduled, which normally takes into consideration its priority, all its operations are sequentially scheduled.

The scheduling process in ISIS involves a hierarchical constraint directed search in the space of all possible schedules. The complete scheduling task is decomposed into a four layer hierarchy in which constraints are passed between adjacent levels. Constraints passed to lower levels in the hierarchy are relaxable and serve only to guide the search (not to restrict it). The four levels considered to construct a job schedule are:

- level 1 - Job Selection selects a job to be scheduled according to the priorities of the jobs. The algorithm used to calculate priorities is based on the category of the job and its due date. Jobs are scheduled one at a time, in priority order.

- level 2 - Capacity Analysis performs a capacity analysis of the plant to determine the availability of the machines required by the selected job. At this level bottlenecks in the plant are detected so that scheduling decisions at the level 3 may be modified to satisfy time constraints. Capacity based scheduling of a job selected by level 1 is performed by applying a critical path method analysis (CPM - see, e.g., [Willis 85], [Davis & Patterson 75] and [Fendley 68]) to the operations involved in the production order.

- level 3 - Resource Analysis performs a detailed scheduling of all resources nec-

essary to produce an order. It extends the scheduling of level 2 by considering more detailed information about operation resource requirements in addition to machines and additional constraints such as preferences, physical constraints, etc. The majority of the search takes place in this phase, essentially beam search.

- level 4 - Reservation Selection selects and assigns reservations for the resources required in the schedule of the job.

ISIS was particularly efficient at reducing inventory levels but it revealed problems in terms of optimising utilisation of bottleneck resources.

SOJA [Le Pape 85] is another constraint-based scheduler, similar to ISIS. It also uses a *job based perspective.* Its main improvement in relation to ISIS is at the job selection level, considering intra-job and inter-job relationships.

RESS-II (REinforcement Scheduling System-II) [Liu 88] was developed by Liu at the University of Edinburgh during the mid 1980's. This scheduling system is similar to ISIS. It also views the shop as a constraint model. However, RESS-II adopts a *resource based perspective.* The important contribution of RESS-II is the utilisation of higher level information to guide the scheduling generation. A central problem that RESS-II attempts to solve is how to predict the effects of local decisions in the global schedule in order to get a good overall resource allocation. RESS-II uses capacity plans to predict bottlenecks and, based on the bottlenecks, to focus on these problems first and continue the scheduling until all the bottlenecks are removed. Basically the strategy adopted in RESS-II consists of two levels of planning: 1) *the reinforcement level* and 2) the detailed planning level. The underlying idea of this strategy is to consider the *reinforcement level* as a pre-planning level to get a global view of the forthcoming events. At this level, the system chooses a critical resource and builds a rough utilisation plan (*reinforcement plan*). Then, at the detail planning level the schedule is generated from scratch but the reinforcement plan is used as a guideline for the decision making process.

OPAL [Bensana *et al* 88] was developed in the late 1980s, at the Centre d'Etudes et de Recherche de Toulouse. OPAL is another constraint-based system. Its main originality has to do with the way the search is performed. The search process is globally depth

first type. However locally it is best first because it has a specific module that always solves the "best conflict". Another feature of OPAL system is the use of "if ... then ..." rules integrating fuzzy set theory model, for empirical and practical knowledge. In order to efficiently integrate and formalise different knowledge sources OPAL system uses different conceptual entities: (1) semantic nets for the description of the problem; (2) precedence graphs and Gantt diagrams for schedule representation; (3) "if ... then ..." rules integrating the use of fuzzy set theory model for empirical and practical knowledge (4) Prolog as the constraint propagation mechanism for the implementation of constraint analysis.

OPIS (OPportunistic Intelligent Scheduler) [Smith *et al* 86] [Smith 87], like ISIS system developed at CMU, represents another qualitative improvement in terms of scheduling systems. OPIS interleaves a *job-based perspective* with a *resource based perspective.* The *resource based perspective* is used to schedule bottleneck resources and the *job-based perspective* is used to schedule non-bottleneck resources. The underlying idea in adopting such an approach is that conflicts or opportunities should be addressed by the most appropriate perspective during the scheduling process rather than relying on the initial detected bottlenecks. In OPIS the idea of bottleneck is pushed one step further compared to other approaches that used that notion, as it was recognised that new bottlenecks can appear during the schedule generation. This ability to detect the emergence of new bottlenecks during the construction of the schedule has been named *opportunistic scheduling.*

Additionally to combining a *job-based perspective* with a *resource based perspective,* OPIS integrates two approaches: *predictive scheduling* techniques and *reactive scheduling* techniques. *Predictive scheduling* can be defined as follows [Smith 87]. Given: 1) a set of production objectives; 2) the state of the factory at the point at which the schedule is called for; 3) a predictive model of how the factory will operate over a specified time horizon - the *predictive scheduling* consists of scheduling decisions made so as to bring about the best possible future state affairs. However, the effectiveness of any *predictive scheduling* technique as a means of coordinating production is limited by the unpredictability of factory operation due to the high dynamic nature of the factory

floor. The ability to *reactively* manage the schedule as unanticipated changes in the factory status occur is vital to a good performance of the factory. The approach adopted in OPIS considers a common viewpoint of *predictive* and *reactive scheduling* techniques. Both types of techniques are considered as an opportunistic problem solving process focused by the current scheduling constraints.

The OPIS architecture is essentially that of a blackboard system [Hayes-Roth 85]. Briefly, a blackboard system is characterised by a central data structure called *the blackboard*, which is often divided into regions or levels. A collection of independent processes called *knowledge sources* (*KS*s) may read and write one or more levels of *the blackboard*, under the supervision of a *control system*, which may be a synchronous global scheduler, a system of concurrency locks or a collection of integrated control-knowledge sources. In this case the blackboard is used for both problem solving and control [Bond & Les Gasser 88]. The blackboard architecture of OPIS is motivated to dynamically focus the scheduling effort according to the current problem constraints, i.e., the most critical perspective. It consists of *the blackboard*, a set of *KS*s (called "Strategic Alternatives"), a scheduling *KS* (called "Search Manager") and a control framework that combines a constraint propagation and a consistency maintenance mechanism with heuristics that take into account the current solution constraints to address specific (re)scheduling strategies. The blackboard represents the current schedule. At the top level, the "Search Manager" co-ordinates the efforts of the *KS*s of the lower level, the "Strategic Alternatives". The "Strategic Alternatives" include two general scheduling methods: the Order Scheduler and the Resource Scheduler, two schedule revision methods: the Scheduler Shifter and the Demand Swapper, and a single analysis method: the Capacity Analyser. The Order Scheduler is based on the constraint-direct search used by ISIS and the Resource Scheduler is a dispatch rule system employing a collection of different heuristics.

The responsibility for planning and coordinating the scheduling actions to be taken in response to a given schedule problem is assumed by the "Search Manager". The "Search Manager's" queue of pending subtasks constitutes its current plan for solving the existing scheduling problem. Both the introduction of factory status changes

and the execution of scheduling tasks by the scheduling *KS*s yields changes to the current factory schedule. These changes are integrated into the current schedule by the constraint propagation mechanism (a schedule maintenance system) which exploits the temporal and capacity restrictions in the factory model to create new constraints. Conflicts and opportunities are identified through this constraint propagation mechanism. The "Search Manager" is informed of the results of the *KS*s activity and schedule changes via posting of "control events". Control events are originated from both externally initiated scheduling updates and internally initiated scheduling actions. The "Search Manager" responds to posted events through the application of a set of "event processing heuristics" which results in the formulation of new scheduling tasks. The "Search Manager's" queue of pending tasks is updated accordingly. Each scheduling task requests a particular analysis, extension or revision relative to the current factory schedule and designates a particular scheduling *KS* to carry out the task. The scheduling proceeds via the formulation and initiation of scheduling tasks. The scheduling process is the implementation of a reactive control capability assigned to the "Search Manager" which results in a continuous revision of its "scheduling plan" as the consequence of the *KS*s execution or external changes.

SONIA [Collinot *et al* 88] was developed at the Laboratoires de Marcoussis. SONIA was developed to integrate predictive and reactive capabilities into the same system, to add reactive capabilities to the SOJA system also developed at the "Laboratoires de Marcoussis". SONIA has several similarities with OPIS. It uses a multiperspective scheduling approach, it combines prediction and reaction and it uses a blackboard architecture. Considering the complexity of complete constraint propagation, which guarantees consistency but it is very time consuming, SONIA developed an interesting constraint propagation mechanism. This mechanism is a "flexible" propagation system with an interpreter that uses axioms consistently with control rules to collectively specify what is expected from the propagation system. By dynamically setting the control rule the system can be adjusted to suit the behavioural needs of the propagation system. Inconsistencies detected are passed over to subsequent analysis. Additionally, SONIA uses a more sophisticated blackboard architecture. Particularly it has a domain blackboard and a control blackboard and it is better provided with analysis

components, particularly a conflict analyser component. However this component is still under development.

A very interesting alternative approach for dealing with the emergence of bottlenecks during the scheduling process has been proposed by [Adams *et al* 88]. This approach, known as *Shifting Bottleneck Procedure* (SBP), consists of an approximation method for solving the minimum makespan problem of the job shop scheduling. It sequences machines one by one, successively, taking each time the machine identified as a bottleneck among the machines not yet sequenced. Every time after a new machine is sequenced, all previously established sequences are locally reoptimised. Both the bottleneck identification and the local reoptimisation procedures are based on repeatedly solving certain one-machine scheduling problems using an optimal algorithm. SBP has allowed for the production of schedules with near-optimal makespan for problems with up to 500 operations. Attempts to generalise the procedure to account for due-dates and more complex objectives seem to have been less successful so far (e.g., [Serafini *et al* 88]).

MICRO-BOSS was developed at CMU [Sadeh 91]. The approach adopted in MICRO-BOSS has been named as *micro-opportunistic* scheduling. Basically, it means that each operation is considered an independent decision point. Any operation can be scheduled at any time. MICRO-BOSS combines a probabilistic model with the constraint satisfaction problem to formulate the search space. Within this model, measures of the reliance of an operation on the availability of a reservation (reservance reliance) and the degree of contention among unscheduled operation for the possession of a resource over some time interval (resource contention) are used to identify critical operations and promising reservations for the operations.

PCP (Preference Capacity Plan) is a predictive model for satisfying conflicting objectives [Berry 91] [Berry 92]. This model aims to satisfy, simultaneously, several conflicting objectives that occur in scheduling environments considering probability and uncertainty theory. PCP assumes that the scheduling problem is basically a problem of resource contention. The idea is to build for each resource an individual preference capacity plan that represents the predicted demand for its capacity considering preference constraints and the dynamic and uncertain nature of the environment. The

information contained in the preference capacity plans is used to focus the attention of the scheduler onto the most highly constrained parts of the problem and to provide information about the satisfaction of objectives and about the impact of constraint relaxation.

[Sadeh 91] and [Berry 91] introduced a fine-grain opportunistic scheduling perspective in which attention focusing is directed by the identification of periods of high resource contention (bottlenecks) but considering the individual operation level. The degree of resource contention is estimated by aggregating assumed operations demand. Bottlenecks are managed by allocating operations contributing to the predicted bottleneck to times outside of the bottleneck period. [Sadeh 91] and [Berry 91] propose a similar approach to the satisfaction of multiple objectives considering the explicit representation of factory objectives within a profile of operation demand over time.

Another qualitative change in scheduling systems is related to Distributed Systems.

YAMS (Yet Another Manufacturing System) [Parunak 87] is an example of a decentralised system. YAMS is a scheduling system whose application domain is flexible manufacturing systems. YAMS can control not only localised set of operations but several factories widely separated from one another. The need to control operations in real time and also to coordinate actions of each machine and factory forces that each plant (and in most cases each machine) has its own computer. YAMS is a distributed system and it is asynchronous. To partition tasks this system uses a negotiation protocol like the contract net described by [Davis & Smith 83]. Briefly, the contract net uses a metaphor of *negotiation* among agents to transfer control on a distributed system. The net consists of a set of nodes that negotiate with each other through a set of messages. Nodes represent the distributed computing resources to be managed. In any given transaction nodes can be categorised into three classes: (1) the "manager" is the node that identifies a task to be done, and assigns it to another nodes for execution; (2) the "bidders" are nodes that offer to perform a task; (3) the "contractor" is a successful "bidder". The more interesting feature of YAMS is the way it adapts the contract net metaphor to the specific nature of the factory floor. Instead of broadcastings tasks to all nodes like in the pure contract net protocol, YAMS uses a hybrid strategy called

"audience restriction". This strategy takes into consideration the specific characteristics of each node to decide whether "to bid or not to bid". Also, to balance the local perspective of the pure contract net protocol, YAMS distributes copies of the global schedule to all nodes. The global schedule is generated using a conventional technique such as MRP or OPT. The copies of the global schedule are distributed to all nodes in the net as an attempt to reduce three types of anomalies: (1) temporal ignorance, (2) spatial ignorance and (3) loading ignorance.

DAS (Distributed Asynchronous System) was developed at the University of Strathclyde [Buchanan *et al* 88], [Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89]. DAS is logically distributed but implemented in a sequential machine. The structure of the DAS is a hierarchy of units: (2) at the top level (strategic), a single unit represents a holistic view of the scheduling problem; (2) at the middle level (tactical), a unit represents an aggregation of similar resources; (3) at the lowest level (operational), a unit represents an individual resource. Each unit within the hierarchy has an attached agent - the Strategic Agent is at the top level; Tactical Agents are attached to the Tactical Units; Operational Agents are attached to the Operational Units. The scheduling problem is decomposed across the hierarchy of communicating agents. The system is asynchronous and can be distributed (one processor per agent). Each agent corresponds to a distinct software process, all of which may run concurrently. The several agents view their scheduling problems as dynamic constraint satisfaction problems. The external world is treated as an agent with one exceptional property, negotiation is not allowed. Because of the asynchronous nature of the decision making process, the system relies heavily on the communication facilities. Additionally, the constraint propagation mechanism provides a basis for the passage of messages across the hierarchy. There is a priority mechanism that allows individual agents to maintain beliefs that are temporarily inconsistent with the global hypothesis. DAS does not differentiate between prediction and reaction.

In DAS, the scheduling process can be synthesised as follows. When a new job is introduced into the system the Strategic Agent, responsible for the whole factory, is notified. The Strategic Agent delegates the work to individual Tactical Agents responsible for

an aggregation of similar resources. The Tactical Agent delegates its operations to Operational Agents, responsible for individual resources. The Operational Agents are responsible for assigning individual times to the operations to be performed on their resources. Whenever an Operational Agent is faced with an over constrained problem it computes an explanation of why it believes the problem is over constrained (an intra-resource conflict set) and notifies its superior Tactical Agent. If a Tactical Agent has an over constrained problem it notifies the superior Strategic Agent by sending it an inter-resource conflict set. The Strategic Agent's mechanisms to solve conflicts are backtracking and temporal constraint relaxation.
The framework proposed in this thesis is inspired by DAS. Chapter 8 presents a more detailed description of DAS as well as a comparative analysis of it with the framework proposed in this thesis.

The CORTES [Fox & Sycara 90] project is an integrated framework for production planning, scheduling and control. It assumes four explicit hypothesis: the generality hypothesis states that there exist a single approach that can optimise decision making across a variety of Planning, Scheduling and Control problems (PSC) without the need of considering a specific approach for each particular production environment; the flexibility hypothesis states that the same approach can be used for both planning, predictive scheduling and reactive control, contrasting with the traditional view of considering a particular approach for each phase of the PSC process; the uncertainty hypothesis states the need to reason about uncertainty in order to anticipate its effects; the scale hypothesis states that large problems should be solved in a qualitatively different manner from small PSC problems. According to this hypothesis, in large problems the aggregate behaviour of the system should be optimised, as opposed to individual entity or job.

CORTES is a distributed system comprising the following modules: the Scheduler; the Planner; the Uncertainty analyser and the Factory model. The underlying paradigm to solve the planning and scheduling problems is Constrained Heuristic Search (CHS). CHS combines the process of constraint satisfaction with heuristic search. The CHS model is a problem space, composed of states, operators and an evaluation function.

Additionally, and this represents an innovation to the traditional definition of a problem space [Newell & Simon 76], the definition of the problem space is complemented with:

- Problem Topology - a structural characterisation of the problem. The problem is represented through a graph, composed of variables and constraints. Variables can be finite/infinite and continuous and discrete. Constraints are n-ary predicates over variables vertices. Constraints can be represented in a conjunctive normal form. The authors consider the possibility of distinguishing topologies that lead to significant changes in problem solving quality (decomposability of graphs into unconnected or loosely connected subgraphs, graph width, etc).

- Problem Textures - features related to the goals of the problem that allow the characterisation of nodes or subgraphs of the problem. The textures allow the search to be focused opportunistically. The textures considered are : value goodness (variable), constraint tightness, variable tightness with respect to a set of constraints, constraint reliance, variable tightness. These textures generalise the notion of constraint satisfiability or looseness defined by [Nadel 86] and apply to both CHS with discrete and continuous variables. The main problem is that most times one does not know all the solutions to the CHS and so texture measures have to be approximated.

- Problem Objectives - rather than define an evaluation function or an an objective function, objectives are embedded directly in the constraint graph so that it can be both propagated and used to make local decisions. (for instance disjunctive constraint sets may have preferences associated with each disjunct; start times for certain variables may also have preferences associated to them).

2.3 Previous Work

Previous work related to job shop scheduling was in the area of airline crew scheduling (e.g., [Gomes 87, Gomes & Almeida 88]). Though airline crew scheduling is substantially different from job shop scheduling, the research carried out in crew scheduling provided some ideas for the current research on the job shop scheduling problem,

particularly in terms of the assignment of times and resources to the operations (see chapter 5).

The Airline Crew Scheduling is the process of assignment of crew to predefined flights, with times, aircraft, and arrival and departure stations defined *a priori*, in such a way that the final schedule assigned to the crew takes into consideration all the legal and contractual constraints and considers the optimisation of the organisational goals. The process of crew scheduling consists of the assignment of technical crew and cabin crew to flights, which have times and aircraft assigned to them *a priori*. In other words, when the crew scheduling process is initiated, it is already known the exact aircraft and the exact times assigned to each flight. It only remains unknow what crew is going to perform which flights.

The most popular formulations of the crew scheduling problem are the set covering and the set partitioning approaches (e.g., [Minoux 85]). However, though these approaches constitute well defined mathematical formulation of the crew scheduling problem, due to its "intractability", the adoption of such approaches hasn't been very successful. The main reason for the relative lack of success of the application of such approaches lay in the fact that the pioneers in the utilisation of such approaches were very large airlines, with very large crew scheduling problems, but especially because the initial models tried to tackle the whole problem as a single set covering or set partitioning problem (e.g., [Arabeyre *et al* 69] and [Roberts 86]). However, for the last few years some successful applications of the set covering and set partitioning approaches have been reported. The main reason of the success of such applications lay in the new approach adopted. The global crew scheduling problem was decomposed into several sub-problems. So, instead of trying to solve a single large problem with a set covering or set partitioning formulation, several smaller and partial set covering or set partitioning problems are considered ([Arabeyre *et al* 69], [Gerbracht 78], [Gerbracht 85], [Marsten *et al* 79] and [Marsten & Shepardson 81]). It is an example of a heuristic decomposition of a problem to cope with its "intractable" nature.

Successful ways of solving the crew scheduling problem include linear programming models (e.g., [Marsten & Shepardson 81] and [Gerbracht 85]) combined linear pro-

gramming with network models, in particular the column generation technique to solve the linear relaxation of the set covering problem (e.g., [Minoux 84]), the graph partitioning approach (e.g., [Gomes 87, Gomes & Almeida 88], [Ball & Roberts 85] and [Ball *et al* 81]) and the network flow problem (e.g., [Zob 79]). A good review on the crew scheduling problem can be found, e.g., in [Bodin *et al* 83], [Carraresi & Gallo 84].

2.4 Potential Research Areas in the JSSP Field

This section is an attempt to highlight some potential research areas in the JSSP field. A list of inter-dependent topics is presented.

- "Intrinsic Complexity"
  - Algorithms - this is a huge area of research and quite "hard", particularly if one aims at finding out polynomial time exact algorithms for a combinatorial problem like the JSSP. This area includes the research on exact algorithms and on heuristic algorithms, particularly polynomial time heuristic algorithms.
  - Evaluation of algorithms - this area includes the evaluation of the algorithms in terms of the quality of the solution (e.g., how far is the solution from the optimal?) and time performance of the algorithm.
- "Extrinsic Complexity"
  - Objectives and constraints - considering a real world environment, the JSSP is characterise by several goals and constraints, usually interacting in conflicting and unforeseen ways, most of them with a qualitative and subjective nature. The identification of the different scheduling objectives and constraints and the ways to integrate them in a scheduling framework constitute a vast area of research.
  - Integration with other functions - the JSSP cannot be seen as an isolated problem within a company. On the contrary, it is vital to the company's "survival" that a harmonious integration exists with other management

functions and with the "external" world. This is an area that needs a lot of research. It has to do with the integration of the JSSP with other functions of the company, particularly other production functions and marketing functions (e.g., setting delivery dates, capacity planning, lot sizing, maintenance strategies, inventory policies etc). These aspects are completely absent in most of the schedulers, particularly in the AI systems. Per se, each of the examples is a topic of research.

- Integration of multiple perspectives and conflicting objectives - another area for research is related to the scheduling perspectives and objectives. Most of the systems adopt either a *job based perspective* or a *resource based perspective* or both, which is designated by *opportunistic scheduling.* More recent systems adopted an *operation based perspective.* Why not generalise the scheduling perspectives by considering other tensions focuses within the shop floor and in the interaction between the shop and other areas of the company? Also, why not allow individual perspectives (individual from the point of view of each job, resource, *etc*) to autonomously manifest themselves, even if that results in a conflicting interaction, as a way of better evaluating conflicts and opportunities as a consequence of the scheduling process. Combining different skills or different forms of tackling a problem is another form of adopting a multiperspective approach.

- Reaction vs. prediction - considering a very dynamic and uncertain environment, it is important for a company to increase its flexibility, its reaction capabilities to the continuous changes in the environment, but at the same time, and in order to minimise risks, to increase its planning and control capabilities. These aspects are crucial and can be formulated as major requirements of a scheduling system– the two sides of the same coin: predictive vs. reactive capabilities. Also, providing a system with reactive features should not mean to ignore the predictive features. Major topics of research are: how to provide and reinforce reactive and predictive capabilities in a scheduler?; how to integrate prediction with reaction in a scheduling system?; are the scheduling objectives different depending on considering a predictive or

a reactive perspective.

- Knowledge acquisition, learning and adaptation - a different area of research is related to the adaptive capability of the system: the integration of the human scheduler knowledge (criticisms and suggestions) and the integration of past scheduling experiences into the system. Much research work needs to be done in terms of knowledge acquisition, learning and adaptation, in the context of the JSSP in particular.
- Satisfaction vs. optimisation - most schedulers aim at satisfaction rather than seeking optimisation, particular in the AI community. On the other hand, for long time the OR community has concentrated efforts on optimal solutions which normally result in systems very dependent on the particular features of each individual problem, normally very simple cases, and even with those limitations the algorithms used require large computational resources. A compromise between the two attitudes seems to be required. Another way of formulating this problem is to combine optimal models for well structured parts of the system while more heuristic type models are used for the less well structured parts of the system or for the components for which efficient optimal models haven't been found.
- Evaluation of the performance of systems - this aspect is closely related to the previous one. It has to do with the evaluation of the performance of the system in terms of: the quality of the solution; the robustness of the solution; the time performance of the system.

2.5 Summary

The job shop scheduling problem (JSSP) is defined in this chapter. The reasons why the JSSP is a "hard" problem are discussed. A succinct review of the current state of the art of job shop scheduling and related work are presented. Potential research areas in the JSSP field are highlighted. In chapter 8 a detailed comparative analysis of the framework proposed in this thesis with existing systems is presented.

Chapter 3

The Underlying Ideas for a Scheduler

In this chapter the major targets of the research reported in this thesis are presented. Some fundamental issues underlying the approach adopted are discussed. The problem solving process is analysed from the point of view of the operators and functions involved in the process. The pros and cons of a distributed problem solver approach are discussed. Two central aspects related to a distributed problem solving approach are highlighted: *Global Coherence* and *Conflict*. The benefits from merging Artificial Intelligence and Operations Research techniques to solve the Job-shop scheduling problem are pointed out. The concept of Structural Awareness is introduced in this chapter.

3.1 The Process of Generating a Solution To a Problem

When we face a problem (*the original problem*) we would like to get the solution to this problem (*the original problem's solution*). However, finding the solution to some problems can be a difficult task.

Figure 3.1 synthesises the process of finding a solution to a given problem. In this figure, the decomposition of the process of generating a solution to a problem into "operators" is emphasised. Given a problem, we would like to know operator A that maps *the original problem* into its solution. However, in real world problems we don't know operator A and therefore to get a solution to a particular problem instead of

using the shortest path in the figure (from *the original problem* to *the original problem's solution*) we are forced to follow the longest path in the figure (a specification of the problem, a solution to the specified problem and maybe an evaluation of the solution to this problem).

But, why don't we know operator A? Let us consider the Job Shop Scheduling Problem (JSSP) explicitly. Operator A is unknown for several reasons. As pointed out in chapter 2, the JSSP is a "hard" problem. In that chapter the "hard" nature of the JSSP was decomposed into two types, the *intrinsic* and *extrinsic* complexity respectively. Let us first consider only the *intrinsic* complexity of the JSSP, i.e., its complexity in terms of solving the problem once it is formulated. The JSSP is a NP-hard problem and so, even assuming that a mathematical formulation of the *the original problem* could be found, the current state of the art would not be able to solve this problem optimally for real world problem dimensions, in a "reasonable" amount of time. Now, let us also consider the *extrinsic* complexity of the JSSP. This is due to the "ill-structured" nature of this problem, with the inability that we have to represent and formulate all the causal relationships among objects that occur in the real world. We refer to this complexity of real world problems through attributes such as "uncertain", "dynamic" and "multiobjective". This type of complexity was underestimated by the OR community for several years, or to be more precise, the OR community has concentrated much more effort on solving the *intrinsic* complexity. However, the *extrinsic* complexity of the JSSP is the main reason for the lack of knowledge about operator A, if we consider "small" JSSPs.

To tackle a real world problem, *an estimator* of operator A has to be obtained. The combined operators, *specifier* and *solver*, represent *an estimator* to the unknown operator A. The operator *evaluator* represents *an estimator* to the unknown operator B. The understanding of this process is crucial. Particularly, three major aspects should be pointed out. When we are solving a problem we should be aware that we are not solving the *original problem* but a *transformed problem.* In other words, it is important to stress that the operator *specifier* performs a *transformation* into the *original problem.* The "distance" between the *original problem* and the *transformed problem*

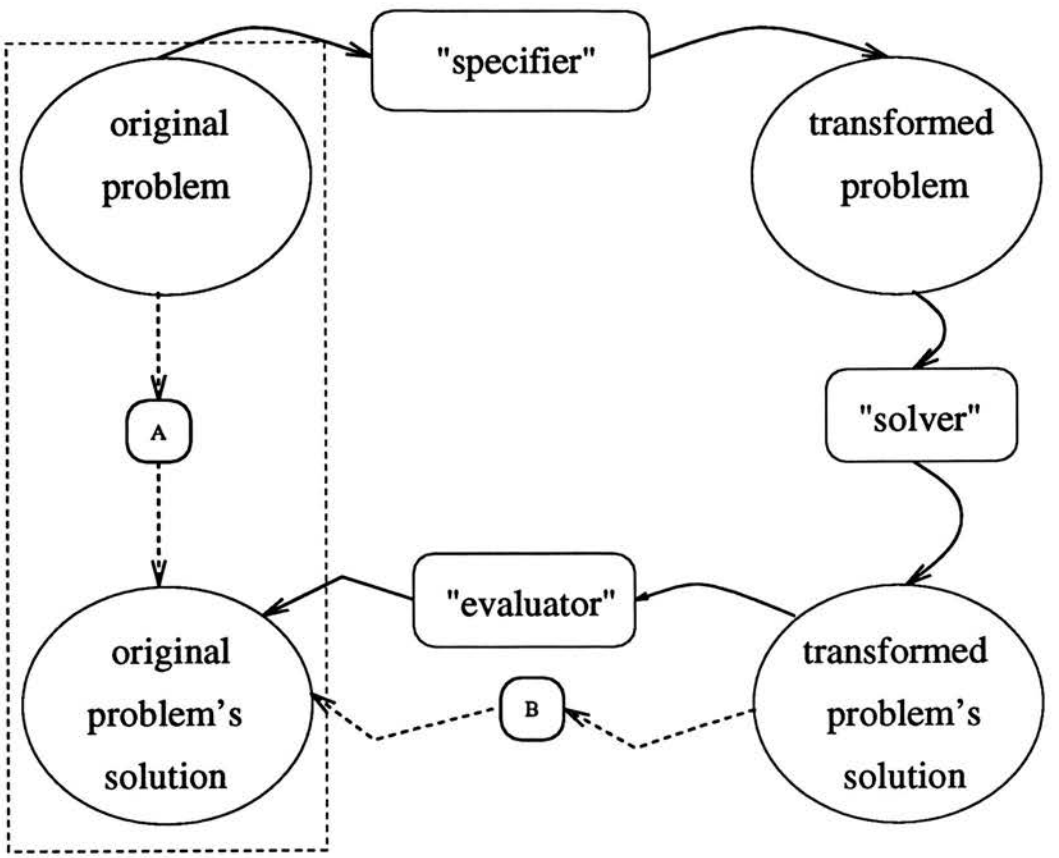

Figure 3.1: The Solution Process Operators

depends on how ill-defined the *original problem* is and also on the "goodness" of the operator *specifier*. An estimator to the operator B should be obtained to measure the distance between the *original problem's solution* and the *transformed problem's solution*. A second aspect is that the *specifier* defines "boundaries" to the *original problem*. A poor definition of the boundaries of the *original problem* can have very bad consequences. Finally, it is important to stress the strong inter-relationship between the operators *specifier* and *solver*. In fact, the *solver* operator is heavily dependent on the way the problem is specified, because it is the set of specified objects, their attributes and relationships among objects that provides a basis for the process of solving the problem. Two different specifications of the same problem result into two different processes of solving the problem and probably give different results. The converse is also true, i.e., depending on the solving process different specifications of the same problem can be formulated. When developing an automatic framework to solve a problem, it is crucial to be aware of these aspects. *A system to generate a solution to a problem should incorporate not only (part of) the operator solver but also (part of) the operator specifier and (part of) the operator evaluator.* There should exist feedback from the *specifier* to the *solver* and vice versa[1]. Automatic systems essentially perform tasks associated with the operator *solver* while little work has been done to provide systems with some ability to (re)specify problems and to evaluate their solutions.

Figure 3.2 gives another perspective on the process of generation of a solution to a problem. In this figure, emphasis is given to the functions performed to get a solution to the *original problem*. The specification function is performed by the operator *specifier*. The *solver* is responsible for analysing the problem (analysis function), for generating a plan to achieve a solution (goal definition and task planning) and finally for implementing the plan (execution). The operator *evaluator* is responsible for the evaluation of the solution. Automatic schedulers normally concentrate their efforts on the execution function. The other functions (specification , analysis , goal definition, task planning and evaluation) are mainly performed by the designers of the automatic

[1]Of course the designers of a system are responsible for the major part of operator *specifier* and even for a great part of operator *solver*. This situation is far from being altered particularly in what concerns the operator *specifier*.

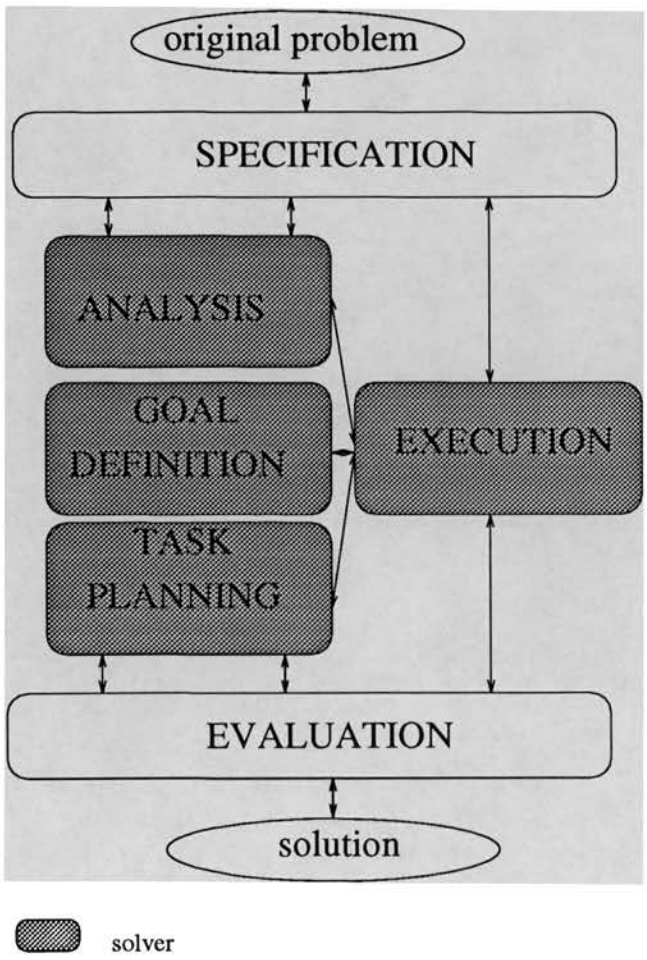

Figure 3.2: The Solution Process Functions

problem solver because of the complexity of these functions. *A major thesis of this research is that a scheduling framework should explicitly incorporate specification, analysis, goal definition, task planning and evaluation capabilities.*

3.2 Distributed Problem Solving

A distributed problem solving approach is analysed in this chapter. Distributed systems are compared to Organisations. Pros and cons of a distributed problem solving approach are analysed. The issues relative to centralised vs. decentralised solutions are outlined.

3.2.1 The Concept

A Distributed Problem Solver (DPS) is defined as a decentralised and loosely coupled collection of semi-autonomous problem solving agents that perform sophisticated problem solving and cooperatively interact to solve problems. The term decentralised means that control, data and knowledge are logically and sometimes physically distributed. Loosely coupled means that individual agents spend more time in computation than in communication. Each agent (also known as actor, knowledge source or node processor) may be embodied in a distinct processor node (but not necessarily). Agents are logically independent from one another and their cooperation is required to achieve their common high level goal [Hern 88, Durfee 89]. At a conceptual and logical level, analysing a scheduling framework from a problem solving point of view corresponds to examining its structure. At the conceptual level, the issues of particular interest relate to modularity, conceptual clarity and simplicity of the design of the system. At a logical and physical level, the issue of primary interest is that of the system performance.

3.2.2 The Organisation Metaphor

A distributed system is a collection of agents and can be viewed as an organisation ([Malone & Smith 84], [Fox 88]). An organisation is defined as a composition of structure and regime control. The set of possible structures range from strict hierarchies

to heterarchies. The design of a distributed system requires the selection of a particular organisational structure and control regime. The particular organisation selected can provide a framework to the agents that composes the system prescribing roles (functional or another aggregate perspective, e.g., products and markets) connected in relationships of authority, communication control and information flow. [Fox 88] considers five major types of organisational structures. The following sections present those different types of organisational structures along with examples of scheduling systems relating to them.

Single person organisation

A single person is the simplest form of organisational structure. The single person performs all tasks, reacting to information and to the environment when necessary. Event-based dispatch rule schedulers are examples of systems belonging to this category.

Group organisation

Some form of group organisation may be needed when the task requires more resources (physical or mental). The organisation is a group of individual members sharing the same information and having a common goal. Each individual member may have different skills and coordination is achieved through mutual agreement upon decisions. More complex control and more information processing becomes necessary. ISIS [Fox & Smith 84] can be considered to be in this category.

Simple hierarchy

When collective decision-making becomes costly within a group, it might evolve to a simple hierarchy. A simple hierarchy has two levels. The top level has a single decision-maker coordinating the efforts of the individuals at the lower level. Only the top level has complete information about the task. The top level is responsible, and has the authority, for effecting changes in the organisation's behaviour. Proper coordination

and distribution of information is required for the organisation to be effective.
The OPIS architecture is a simple hierarchy. It consists of a blackboard, a set of Knowledge Sources (the "Search Manager" and the "Strategic Alternatives") and a control framework that combines a constraint propagation and a consistency maintenance mechanism. At the top level a single decisionmaker coordinates the efforts of the *KS*s of the lower level. The *KS*s of the lower level are "Strategic Alternatives" which include two general scheduling methods (Order Scheduler and Resource Scheduler), two schedule revision methods (Scheduler Shifter and Demand Swapper) and a single analysis method (Capacity Analyser).

Uniform Hierarchy

A uniform hierarchy may be defined where it becomes unwieldy for a single decision maker to be responsible for all the control. The next form of organisation occurs when the decisionmaker is incapable of processing all the information. Multiple levels of management are created to ensure proper and decentralised decisionmaking. Each level of the hierarchy acts as a filter on the information and decisions that are propagated up and down the hierarchy.
DAS[Buchanan *et al* 88], [Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89] is an example of a system with a uniform hierarchy with three levels of management. Each level of the hierarchy acts as an information filter both up and down the hierarchy.

Multidivisional hierarchy

In a multidivisional hierarchy, the organisation is split along product lines. Each division is in full control of the tactics involved in producing their product. Strategic control is vested in an elite staff assigned to a general office concerned with strategic planning, appraisal and control including resource allocation. YAMS [Parunak 87] combines characteristics of a multidivisional hierarchy and a market-like structure (see next section).

Market-like Structure

In a market-like system, the notion of a pricing system is introduced. A number of organisations are capable of producing a product or supplying a service. Actions are initiated after the successful negotiation of a contract among agents. This system eliminates all forms of control between units, with all communication being contained in a contract to purchase some product or service. Control is exerted through the price of product. Price reflects the marginal cost of the product. The assumption is that through marginal pricing of goods, all the resources will be utilised without waste. YAMS [Parunak 87] combines characteristics of a multidivisional hierarchy and a market-like structure. To partition tasks this system uses a negotiation protocol similar to the contract net protocol described by [Davis & Smith 83]. Some modifications were introduced in the contract net model to cope with the specific nature of the factory floor. The hybrid strategy adopted in YAMS is called "audience restriction". The main difference of this strategy compared to the pure contract net is that for a given bid not all the nodes are considered as potential contractors.

3.2.3 Pros and Cons of a DPS

The underlying reasons to adopt a DPS approach are summarised below:

- From the *intrinsic complexity* point of view:

  Tractability — the decomposition of the whole problem into smaller and more manageable problems assigned to different agents is a good strategy to tackle the computational complexity of the scheduling problem, in order to transform an "intractable" problem into a set of less "intractable" problems. The strategy is "divide and conquer".

  Speed — one of the main aims when adopting a (physical) distributed approach is to achieve high-speed problem solving. Concurrency may increase the speed of computation and reasoning, especially if the agents do not spend much time communicating with each other.

- From the *extrinsic complexity* point of view:

  Opportunism — the adoption of a set of distinct agents with different skills facilitates and gives the opportunity to apply a variety of perspectives.

  Specialisation — agents (and processors) may be specialised according to the nature of the tasks assigned to the corresponding agent.

  Structure — looking at the system from a distributed point of view enforces the explicit consideration of aspects such as the decomposition of the problem into smaller problems, modelling agents in terms of their skills and their interaction with each other. This encourages conceptual clarity and simplicity of design.

  Geography — a distributed system may be associated with natural geographic distribution.

  Reliability — distributed systems may be more reliable than centralised systems. For example, when a centralised scheduler fails, the entire systems stops, while decentralised systems can continue to operate with all the remaining nodes.

  Cost — if implemented as a network of low-cost computer systems, a distributed system may prove to be cost effective.

Despite the many potential advantages of distributed systems, their design remains problematic. Once data, knowledge and control is distributed, it becomes less clear how to ensure coherent and co-operative behaviour among agents, i.e., how to maintain the *Global Coherence* of the system as whole. [Bond & Les Gasser 88] identify the following criteria to evaluate the *Global Coherence* of a system: solution quality, efficiency, clarity and graceful degradation. Solution quality is the system's ability to reach satisfactory solutions and the quality of the solutions it produces. Efficiency is the system's overall efficiency in achieving an end. Clarity is the conceptual clarity of the system's actions and the usefulness of its representation. Graceful degradation is how gracefully the system degrades in the event of failure or uncertainty. Ensuring *Global Coherence* within a distributed problem solving structure is a crucial aspect, particularly considering the *tightly coupled* nature of factory scheduling problems. Within a DPS environment *Conflict* can arise either due to incompatible constraints of the

problem (in this case we include conflicting situations as a consequence of external events) or due to conflicting decisions performed by the agents (this case includes incompatible situations as a consequence of the solving process) [Bond & Les Gasser 88], [Durfee *et al* 87], [Decker *et al* 89]. It is important to define possible scenarios in which *Conflict* can occur, how to represent and recognise the *Conflict* and how to deal and resolve the *Conflict.*

3.2.4 Centralised vs Decentralised Systems

Earlier, several reasons to adopt a distributed problem solving approach were mentioned. Normally, however, the main reason to use a physically distributed system is to achieve high speed problem solving. In order to achieve speed, situations in which processors get in each others' way must be avoided. In other words, agents should be *loosely coupled*; they should spend more time in computation than in communication. Scheduling problems, however, tend to be *tightly coupled*, which means that decomposing the whole problem into independent subproblems is very difficult or maybe impossible. Nevertheless, considering the benefits of using a distributed approach, several strategies have been adopted to decompose and decentralise factory scheduling problems. Some of these strategies[2] are outlined below.

Product Decomposition — this strategy corresponds to having different agents representing different *products and production processes.* Even when the production processes show some interdependencies, the adoption of a different agent for each product may facilitate an opportunistic approach by product.

Management Functions Decomposition — this strategy corresponds to having different agents with different skills in terms of *management functions*, such as production, inventories, resources, maintenance of resources, purchasing of materials etc. *Resource-based scheduling* and *job-based scheduling* are examples of this category of decomposition.

[2]These strategies can sometimes be combined.

Problem Solving Functions Decomposition — this strategy corresponds to having different agents with different skills in terms of *problem solving* function, i.e., specification, analysis, goal definition, task planning and evaluation. For instance in OPIS the top level manager is responsible for planning and coordinating the agents of lower levels. Also in OPIS there is an analysis method, the Capacity Analyser.

Degrees of Centralisation — this strategy corresponds to assigning different *levels of information and control* to different agents. Typically, agents with more information control and centralise the decisions of agents with less information. Hierarchical structures are an example of this approach. There are several criteria which may be applied in designing a hierarchical structure. In DAS, a Tactical Agent is responsible for an aggregate resource, a set of similar resources, while Operational agents are responsible for individual resources. Tactical Agents distribute operations among Operational Agents. Operational Agents are responsible for individual machines and they only have to assign times to the operations assigned to them by the corresponding Tactical Agent. Another approach is *hierarchical production planning* where the original production planning problem is divided into a hierarchy of subproblems. The upper level deals with strategic decisions for the planning horizon, while the lower level deals with more short term scheduling (e.g., [Hax & Meal 75]). Another criterion for decentralisation is the separation of the decisions regarding lot sizing and lot scheduling.

Contracting — this strategy is used when several agents are capable of performing the *same task.* Task assignment is achieved through *contracting.* YAMS uses a negotiation protocol similar to the contract net protocol.

Common Deductive Database — this strategy corresponds to distributing the decision making process among agents but data is centralised in a common deductive database. The decisions of each agent of the system are public since a public and common database is adopted. A typical example of this type of architecture is the *blackboard system.*

Predictive vs reactive activities — this strategy corresponds to providing some agents with predictive capabilities (normally high level agents with more global information)

while agents of lower levels are provided only with reactive capabilities [Le Pape 91].

As a final remark it is worth mentioning that it is possible to design a system using a DPS approach at a logical level and implement it in a sequential machine. Such a situation implies the existence of a mechanism to enforce the sequential synchronisation of all agents, i.e., the simulation of a distributed system in a sequential machine. Speed gain, one of the most attractive features that one can expect from a physical distributed system, is *a priori* excluded if such an approach is adopted. However, even if we consider only a logical point of view, a DPS approach enforces the explicit consideration of aspects such as decomposition of the problem into smaller problems, modelling agents in terms of their skills and the interaction of agents with each other. Because distributed systems (logically or physically) tend to be highly modular they offer conceptual clarity and simplicity of design. An example of a logical distributed system (not a physically distributed system) is DAS.

3.3 Major Targets of Research

The research reported in this thesis focuses on:

- Integration of multiple scheduling perspectives - it corresponds to the integration of a *job based perspective* with a *resource based perspective*, resulting into an *operation based perspective*. It also consists of integrating empirical knowledge (heuristics and dispatch rules) with more theoretical knowledge (optimal algorithms and more elaborated mathematical models).

- Integration of predictive capabilities with reactive capabilities - the framework selected for a job shop scheduler is inspired by DAS ([Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89]), essentially a reactive system. DAS does not differentiate between reaction and prediction. The research reported in this thesis investigates strategies to reinforce the *predictive* capabilities of a system that is essentially *reactive*.

- Increasing *optimisation* rather than settling for *satisfaction* - the underlying idea is to try to tackle the well defined components of the JSSP with mathematical models, if possible with some guarantees of local optimal solutions, while more heuristic type models are used for the more complex and ill defined parts of the problem.

In chapter 6 and chapter 8 the targets of the research reported in this thesis will be addressed again.

A Distributed Problem Solving (DPS) approach was chosen for a job shop scheduling framework. The framework adopted for a job shop scheduler was inspired by DAS, Distributed Asynchronous System.

Considering the DPS point of view, the research reported in this thesis investigates strategies to ensure the *Global Coherence* and *Conflict Resolution* within a distributed problem solving environment. The approach adopted is to consider *Conflict* as a normal and even "healthy" state within a system, as long as it is "under control". *Conflict* arises as the consequence of the dynamic nature of the environment. Also, *Conflict* can arise if multiple perspectives are used within the system. The first form of *Conflict* is "non-negotiable" and is imposed by the environment. Managing this type of *Conflict* corresponds to providing the system with reactive capabilities. The latter form of *Conflict* arises from the solution process. The idea of allowing *Conflict* is not original, it is also present in DAS. However, in the approach reported here, the idea of *Conflict* is pushed one step further, as it was recognised that *Conflict* as a consequence of the scheduling process should be exploited as a way of allowing the different agents of the system to freely express their interests. *The thesis is that not only is it important to deal with Conflict, Conflict should be exploited to increase Global Coherence.* In othe words, the *conflicts* that occur as a consequence of the scheduling process provide good information in terms of the opportunities and weakness of alternative schedules.

Some inter-related guidelines adopted in the research reported in this thesis are summarised in the following points:

- *The Organisation of the system* - the organisational structure of DAS is essentially a *uniform hierarchy*, with three levels of management. Providing the agents of the system with different functional capabilities was a central guideline to the approach adopted. In particular, analysis, goal definition, task planning and evaluation capabilities were provided to the agents of the system, considering their particular position within the hierarchy. This idea has been partially used in other scheduling systems (e.g., OPIS, SONIA and DAS), though it is more developed in other application domains (e.g., [Hayes-Roth 85],[Durfee 89], [Durfee *et al* 87] and [Decker *et al* 89]).

- *Pluralism* - [Kornfeld & Hewitt 88], introduced the idea of the "scientific community metaphor". The success of scientific research depends heavily on complementary perspectives and knowledge. In the JSSP a generalisation of perspectives can correspond to different "interests" within the shop floor, jobs and resources, but also different roles and skills assigned to the different agents of the system, as well as different sources of expertise such as OR vs. AI techniques and theoretical vs. empirical knowledge. Exploiting "conflicting" perspectives is a form of pluralism.

- *Merge of* OR *&* AI *techniques* - in chapter 2 several considerations were made about OR and AI techniques and about the interest in combining these two approaches. In fact, having in mind the "hard" nature of the JSSP, it is essential to adopt a multidisciplinary approach. A major thesis of my research is that the combination of OR and AI techniques can produce synergetic effects.

  - Why OR techniques?
    * To tackle well structured components of the JSSP
    * Long experience dealing with combinatorial problems and with the JSSP in particular: exact approaches and heuristic approaches. There is no point in re-inventing the wheel.
  - Why AI techniques?
    * To capture the human style reasoning of practical dedicated expertise provided by the shop-floor managers. This knowledge is informal, im-

precise and gained from experience. Its main importance is to incorporate into a scheduling framework the "qualitative" dimension of the shop floor.

* AI techniques are more "knowledge based", more flexible, more general-purpose and therefore more adaptable to real world environments

– why AI and OR techniques?

More efficient solutions for the JSSP can be achieved by combining:

* *Objective* models for those parts of the problem capable of a mathematical description
* *Human Style Heuristic Reasoning* models for the more complex behavioural parts

- *Structural Awareness* - [Bond & Les Gasser 88] introduce the idea of "contextual awareness". They say that "Coherence and coordination can be improved by giving nodes greater knowledge about the context in which they are making decisions - greater knowledge about the goals, plans and activities of other agents, deeper knowledge of the problem domain (which is the context for individual domain-dependent decisions), and greater temporal context (e.g., greater lookahead or history)". Structural Awareness is a form of "contextual awareness". It corresponds to a set of mechanisms assigned to the different agents of the system to make them aware of the structural and intrinsic properties of the (sub)problems that they have to solve and the interaction of their (sub)problems, without relying on communication with each other.

- *Detection/Resolution of conflict* - in order to detect and analyse conflicting situations, it is crucial to provide the agents with analysis, planning and evaluation capabilities.

3.4 Summary

In this chapter the major targets of the research reported in this thesis are presented: the integration of multiple scheduling perspectives; the integration of predictive ca-

pabilities with reactive capabilities; increasing *optimisation* rather than settling for *satisfaction.* Considering the DPS point of view, the research reported in this thesis aimed to investigate strategies to ensure the *Global Coherence* and *Conflict Resolution* within a distributed environment. The underlying ideas to tackle those targets are also presented in this chapter: the assignment of different roles and skills to agents within a distributed problem solver framework; ensuring *pluralism*; the merge of OR and AI techniques; ensuring *structural awareness*; and providing agents with mechanisms to detect and solve *Conflict.*

Chapter 4

An Overview of the Scheduling Framework

An overview of the framework proposed for a job-shop scheduler is presented in this chapter. The overall structure of the system is depicted. A general description of the agents of the system and their functions is presented. The scheduling process is outlined. The concepts of *Conflict* and *Structural Awareness* within the scheduling framework are introduced.

4.1 The Overall Structure of the System

Figure 4.1 represents the overall structure of the job-shop scheduling framework. This structure is adapted from DAS ([Burke & Prosser 89], see also chapter 8).

At the Strategic Level, the Strategic Agent is responsible for the whole problem, particularly for assigning work to the Tactical Level and for detecting and solving conflicts that occur from the scheduling decisions performed by the Tactical Agents. At the intermediate level, the Tactical Level, there are two categories of Tactical Agents: the Resource Tactical Agents and the Job Tactical Agents. The Job Tactical Agents are responsible for the jobs. There are as many Job Tactical Agents as the number of jobs to be scheduled. Each Job Tactical Agent is responsible for assigning time windows[1] to

[1] A time window is defined by: an earliest start time (est); an earliest finish time (eft); a latest start time (lst); a latest finish time (lft)

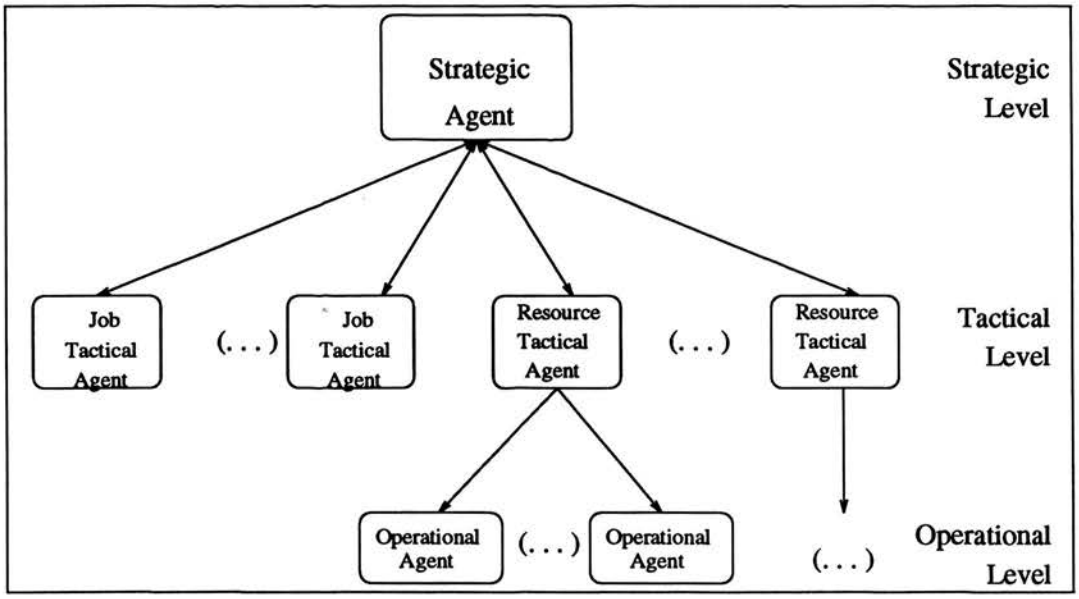

Figure 4.1: Overall structure of the system

the operations of its job, considering the technological constraints between operations of the same job. The Job Tactical Agents perform their scheduling tasks assuming unlimited resources. The Resource Tactical Agents are responsible for the aggregate resources. An aggregate resource is a set of similar machines capable of performing a certain operation. There are as many Resource Tactical Agents as the number of aggregate resources. The Resource Tactical Agents are responsible for scheduling the operations on the different machines of their aggregate resources taking into account the time windows defined by the Job Tactical Agents. The Resource Tactical Agents assign individual machines and start times to its operations. If the system includes Operational Agents, the Resource Tactical Agents send the operations to the corresponding Operational Agent. Each Operational Agent is responsible for an individual machine. The Operational Agents are responsible for local improvement of the schedule suggested by the the corresponding Resource Tactical Agent to the set of operations assigned to them.

4.2 Outline of the Scheduling Process

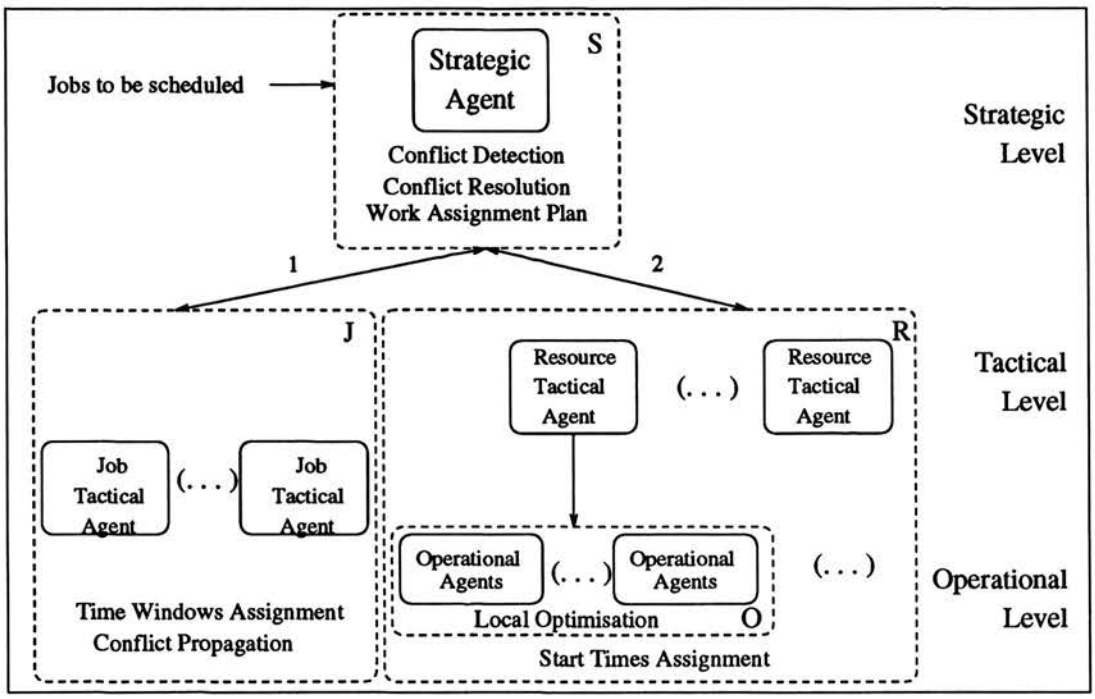

Figure 4.2: Schedule Generation Process

Figure 4.2 depicts the process of schedule generation. The jobs enter the system via the Strategic Agent. The Strategic Agent sends each job to the corresponding Job Tactical Agent for the assignment of time windows to the operations that constitute the job. Time windows define the earliest start time, earliest finish time, latest start time and latest finish time of each operation. The Job Tactical Agents assign time windows to their operations solving a critical path method problem (see e.g., [Willis 85], [Davis & Patterson 75] and [Fendley 68]) and independently of each other. This task is performed assuming unlimited resources. The Job Tactical Agents send the results of the time windows assignment to the Strategic Agent.

Once the Strategic Agent receives the time windows for all the operations of all the jobs, it assigns work to the Resource Tactical Agents. Operations to be performed on the same aggregate resource are sent to the respective Resource Tactical Agent with the respective time windows. The Resource Tactical Agents assign individual machines and start times to the operations to be performed on their aggregate re-

sources. Analogously to the Job Tactical Agents, the Resource Tactical Agents perform their scheduling tasks independently of each other. If the system is provided with Operational Agents, the Resource Tactical Agents assign work to each of their Operational Agents. Each Operational Agent receives the information concerning the set of operations to be performed on its individual machine. The Operational Agents are responsible for locally improving the schedules proposed by their superior Tactical Agent. The Operational Agents carry out their scheduling tasks independently of each other. Operational Agents send their schedules to the Strategic Agent. Note that, if the system does not include Operational Agents, the Resource Tactical Agents send their schedules directly to the Strategic Agent.

The Strategic Agent examines the schedules generated by the Resource Tactical Agents, with the intervention or not of the Operational Agents[2], in order to detect conflicts.

Conflicts are categorised per job and the respective information is sent to each Job Tactical Agent for conflict propagation and for re-assignment of time windows to the operations.

Based on the information returned by the Job Tactical Agents, the Strategic Agent generates a plan for conflict resolution.

According to the plan for conflict resolution, the Strategic Agent re-assigns work to the Resource Tactical Agents.

The Resource Tactical Agents reschedule the operations on their resources considering the new time windows redefined by the Strategic Agent. If the system includes Operational Agents, the Resource Tactical Agents pass on the operations with the respective time windows to the Operational Agents for them to improve the schedule on their resources. The Operational Agents return the improved schedules to the Strategic Agent. If the system does not include Operational Agents, Resource Tactical Agents send their schedules directly to the Strategic Agent.

The Strategic Agent analyses the schedules produced by the Resource Tactical Agents

[2]Hereafter in this chapter, whenever we refer to the scheduling task performed by the Resource Tactical Agents it is assumed that it can be carried out either with the intervention of the Operational Agents or without it, unless the opposite is explicitly stated.

in order to find conflicts. The process continues until a final schedule, without conflicts or with only unsolvable conflicts, is reached.

Schematically, and referring to figure 4.2, the scheduling process can be described in terms of iterations, each iteration involving different sub-problems.

The first sub-problem is associated with the Strategic Agent (box "S"). The Strategic Agent detects conflicts that occur as the result of the assignment of start times by the Resource Tactical Agents. The Strategic Agent sends the detected conflicts to the Job Tactical Agents involved in the conflicts. When initiating the scheduling process, the Strategic Agent sends all the operations to the respective Job Tactical Agent for time windows assignment.

The second subproblem corresponds to the assignment of time windows to operations performed by the Job Tactical Agents (box "J"). This assignment is either a new assignment or a redefinition of existing time windows due to conflicts propagation. Note that there are no links between the Job Tactical Agents since they perform their scheduling tasks independently of each other.

Based on the information sent by the Job Tactical Agents, the Strategic Agent defines a plan for (re)assignment of work to the Resource Tactical Agents. This task planning corresponds to the third sub-problem (box "S").

The fourth sub-problem corresponds to the assignment of start times to operations to be performed by the Resource Tactical Agents (box "R"). This sub-problem might include an optimisation cycle to be performed by the Operational Agents (box "O").

This framework is very suitable for parallel implementation since the different Tactical and Operational Agents perform their tasks independently of each other and so they can perform their tasks concurrently. This issue is further discussed in chapter 10.

4.3 Conflict

The idea of *conflict* (e.g., [Burke & Prosser 89]) within a scheduling framework is not new. The approach adopted is to consider *conflict* as a normal and even "healthy"

state within a system, as long as it is "under control". A major thesis of the research reported here is that not only is it important to deal with *conflict* but also *conflict* as a consequence of the scheduling process should be exploited as a way of integrating different scheduling perspectives, as a way of detecting the opportunities and weaknesses of alternative schedules.

In the scheduling framework presented here, *conflict* as a consequence of the scheduling process occurs due to the adoption of two different perspectives:

- *Order perspective* - the Job Tactical Agents assign time windows to their operations independently of each other. This form of assignment allows each Job Tactical Agent to define the "best" time windows for its operations. However, because the assignment of time windows is performed without considering the resource constraints, *conflict* is very likely to occur.

- *Resource perspective* - the Resource Tactical Agents and the Operational Agents assign start times to the operations independently of each other. This form of assignment allows each Resource Tactical Agent to "optimise" the utilisation of its machines. However, because the assignment of start times to operations is performed without considering the constraints in terms of jobs, *conflict* is very likely to occur. Note that the Resource Tactical Agents take into consideration the time windows proposed by the Job Tactical Agents. However, time windows are used only as a reference. Apart from the indication in terms of job constraints embodied in the time windows, the Resource Tactical Agents are not aware of the constraints between operations of the same job.

The Strategic Agent is responsible for detecting the *conflicts* that occur as the result of the assignment of start times performed by the Resource Tactical Agents. The Strategic Agent is also responsible for coordinating the process for conflict resolution. In order to solve *conflicts*, the Strategic Agent adopts a *micro-opportunistic*[3] approach

[3]The term "micro-opportunistic" does not correspond entirely to the way it is used by [Sadeh 91, Berry 91]. The most important aspect that we want to emphasise by using it is that the Strategic Agent makes its scheduling decisions considering operations individually.

since attention is focused on individual operations. The Strategic Agent takes into consideration that each Resource Tactical Agent generates the schedule of the operations on its aggregate resource considering the most favourable utilisation of the time windows proposed by the Job Tactical Agents. The Strategic Agent analyses the schedules generated at the Tactical and Operational Level in order to:

- detect conflicts among the proposed schedules
- define the "criticality" of each operation
- define goals and (re)plan the tasks to assign to the Tactical Agents

In order to solve the *conflicts* detected and based on them, the Strategic Agent makes a plan for re-assignment of work to the Tactical Level involving three types of decisions mainly:

- which start times assigned by the Resource Tactical Agents should be accepted;
- which time windows should be redefined by the corresponding Job Tactical Agents;
- which operations should be rescheduled by the corresponding Resource Tactical Agents;

The scheduling process terminates when the schedule generated by the different agents of the system does not have any *conflicts* or the existing *conflicts* are unsolvable.

The concept of *conflict* is central to the framework proposed here. In this scheduling framework, Tactical Agents are given the opportunity to formulate their best schedules independently of each other. In this sense, exploiting *conflict* corresponds to allowing each agent to express its own interests independently of each other. Detecting and solving *conflict* corresponds to coordinating and making compatible the conflicting interests of the agents. Considering the importance of this aspect, the acronym EXPLICIT was chosen as the name of the job shop scheduling framework – EXPL O I ting C onfl I C T. Hereafter the acronym EXPLICIT will be used to identify the job scheduling framework

reported in this thesis. The concept of *conflict* will be formally defined in the next chapter.

4.4 Structural Awareness

In the previous section the concept of *conflict* was introduced as a central aspect embodied in EXPLICIT, a normal and even "healthy" state within the system, as long as it is "under control". In such a "conflicting" environment it is essential to provide the different agents of the system with knowledge about the characteristics of the problems they have to solve and about the context in which they make decisions.

> *Structural Awareness* is the knowledge that each agent holds about the structural and intrinsic properties of its own (sub)problem and the interaction of its (sub)problem with other agent's problems, without relying too much on communication with the other agents of the system.

The underlying thesis to make the agents of the system *structurally aware* is that the knowledge about the inherent structural properties of a domain can be used to alleviate its "intractability".

- Objective:
  - to provide the agents of the system with mechanisms to ensure the *Global Coherence* of the system as whole.
- Sub-objectives:
  - to exploit the different topologies of the (sub)problems, to guide the search
  - to decompose the search space into localised search spaces in order :
    * to exploit parallelism
    * to reason about subproblems interaction (conflicts)
    * to cope with dynamic environments

Structural awareness is intimately connected with the idea of contextual awareness associated with distributed systems[4]. EXPLICIT was conceived having in mind the concept of *structural awareness*, in particular in terms of the design of the algorithms assigned to the different agents of the system. The set of mechanisms provided to the agents of EXPLICIT in order to ensure their *structural awareness* are highlighted in chapter 6. The next two chapters contain the detailed description of EXPLICIT.

4.5 Summary

EXPLICIT is the name of the job shop scheduling framework reported in this thesis. EXPLICIT stands for "exploiting conflict".

The overall structure of EXPLICIT was presented in this chapter. The scheduling process was outlined. Two central concepts embodied in EXPLICIT were introduced in this chapter: the concept of *conflict* and the concept of *structural awareness.*

[4][Bond & Les Gasser 88] claim that "coherence and coordination can be improved by giving nodes greater knowledge about the context in which they are making decisions - greater knowledge about the goals, plans and activities of other agents, deeper knowledge of the problem domain (which is the context for individual domain-dependent decisions), and greater temporal context (e.g., greater lookahead or history)".

Chapter 5

Scheduling with EXPLICIT

A detailed description of the scheduling process adopted in EXPLICIT is presented in this chapter. The scheduling entities and scheduling functions assigned to the system's agents are described in detail. Representation issues and algorithms are included in this chapter. The characteristics of the types of problems that can be solved by EXPLICIT are also discussed.

5.1 Major features of the type of JSSPs tackled by EXPLICIT

EXPLICIT is intended to tackle a variety of job shop scheduling problems. The idea is to develop a framework flexible enough to cover a large spectrum of problems. For particular problems with specific characteristics, the system can be provided with more accurate processes. For example, the Operational Agents can be used as a way to tune the system for a particular domain and a particular problem.

The following list contains the general features of the type of problems currently covered by EXPLICIT[1]

- Set of n jobs: $J_1, J_2, \ldots, J_n$. The set of indices of the jobs is denoted by N. Jobs are composed of distinct operations. The set of indices of the operations of a

[1]Though EXPLICIT is conceptually suitable to tackle problems characterised by the features described in this section, the current implementation only considers the problems with the characteristics described in chapter 9, section 9.2.

given job, say job j (J_j), is denoted by N_j

- the order of processing of each job on each machine (operation) does not have to be the same (Job Shop)
- there are precedence constraints between operations of the same job
- parallel plans may occur
- jobs may have a due date or not
- no preemptive jobs
- arbitrary ready times for each job
- In-process inventory is allowed, i.e, jobs may wait for their next machine to be free
- deterministic times, i.e., no *randomness*, in particular:
  * the number of jobs is known and fixed
  * the processing times are known and fixed
  * the ready times are known and fixed
  * all other quantities needed to define a particular problem are known and fixed

- Set of m sets of machines: $M_1, M_2, \ldots, M_m$ (aggregate resources; parallel machines; multiple processors); the set of indices of the aggregate resources is denoted by M. Aggregate resources are composed of individual identical machines. $|M_i|$ denotes the number of individual machines of M_i.
  - $|M_i| \geq 1$
  - The different individual machines of each aggregate resource can perform the same operation; individual machines can have different processing times to perform the same operation
  - The number of individual machines per aggregate resource does not have to be the same for the different aggregate resources
  - Arbitrary ready times for each machine
  - Each machine can only process one operation at a time

- Machines may be idle
- The machines might be unavailable for certain periods (e.g., planned maintenance)
- deterministic times, i.e., no *randomness*, in particular:
  * the number of machines is known and fixed
  * the processing times are known and fixed
  * the ready times are known and fixed
  * all other quantities needed to define a particular problem are known and fixed

5.2 Formulation of the problem

It is useful to represent the whole problem on a graph $\mathcal{G} = (\mathcal{O}, \mathcal{R}, \mathcal{A}, \mathcal{E})$, with node set $\{\mathcal{O}, \mathcal{R}\}$, and ordinary (conjunctive) arc set $\mathcal{A}$ and disjunctive arc set $\mathcal{E}$. Figure 5.1 illustrates this graph.

The node set $\mathcal{O}$ of $\mathcal{G}$, $\mathcal{O} = \{o_{ij} : i \in N_j, j \in N\}$, N the set of indices of the jobs to be scheduled and N_j the set of indices of the operations of a given job j, corresponds to the operations of the jobs to be scheduled (represented by a circle in the graph). The node set $\mathcal{R}$ of $\mathcal{G}$, $\mathcal{R} = \{r_{ij} : i \in M_j, j \in M\}$, M the set of indices of the different aggregate resources or machine types and M_j the set of indices of the individual machines of a given aggregate resource j, corresponds to the different machines on which the different jobs have to be scheduled (represented by a square in the graph). The arc set $\mathcal{A}$ of $\mathcal{G}$, $\mathcal{A} = \{(o_{ij}\ o_{lj}) : i,l \in N_j, j \in N\}$, corresponds to precedence relations between operations of the same job (represented by full arrows in the graph). The disjunctive arc set $\mathcal{E}$, $\mathcal{E} = \{(o_{ij}\ r_{lk}) : i \in N_j, j \in N, l \in M_k, k \in M\}$, denotes the alternative machines where a given operation can be performed (represented by dashed arrows in the graph).

The set of arcs $\mathcal{A}$ decomposes the graph $\mathcal{G}$ into subgraphs (O_j, A_j), where $O_j = \{o_{ij} : i \in N_j, j \in N\}$, $\mathcal{O} = \bigcup(O_j : j \in N)$ and $A_j = \{(o_{ij}\ o_{lj}) : i,l \in N_j, j \in N\}$, $\mathcal{A} = \bigcup(A_j : j \in N)$. Each subgraph (O_j, A_j) corresponds to a job j, $j \in N$. Figure 5.2

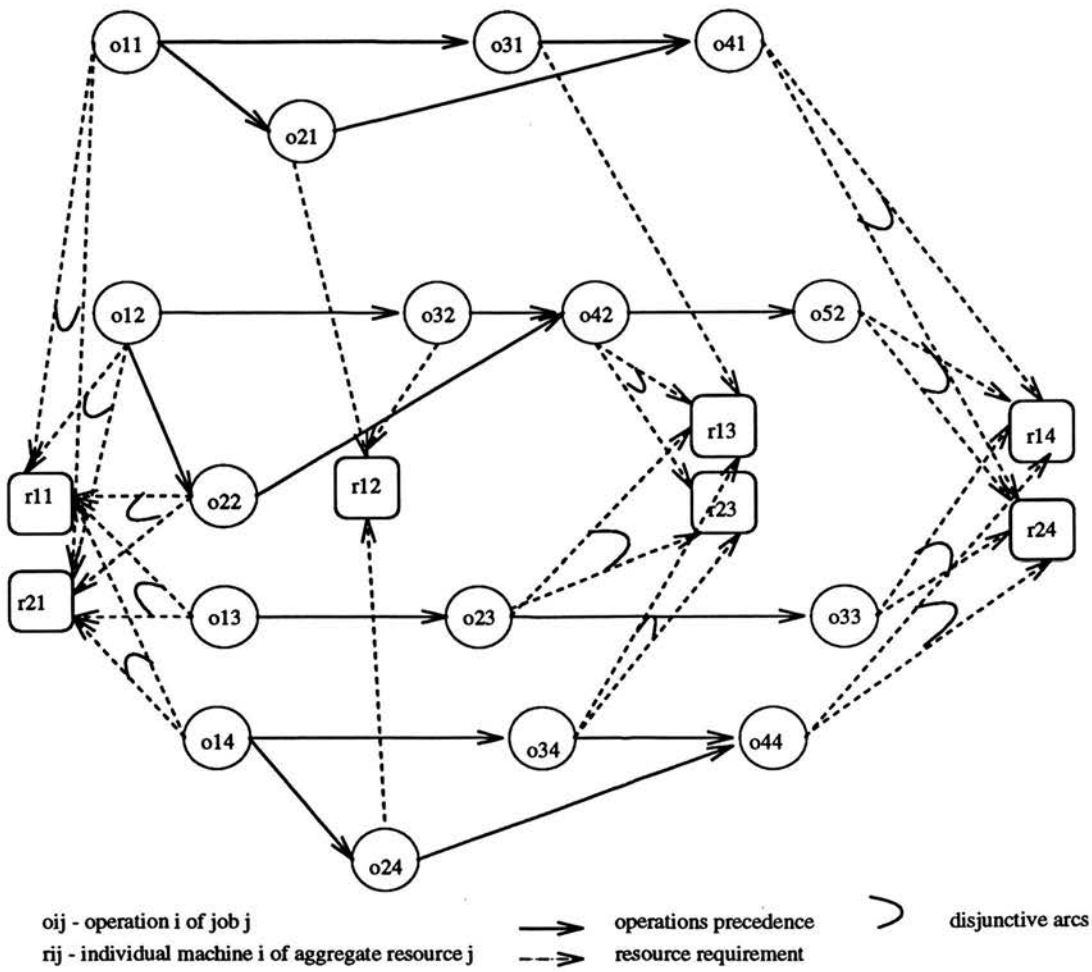

Figure 5.1: The representation of the job shop scheduling problem

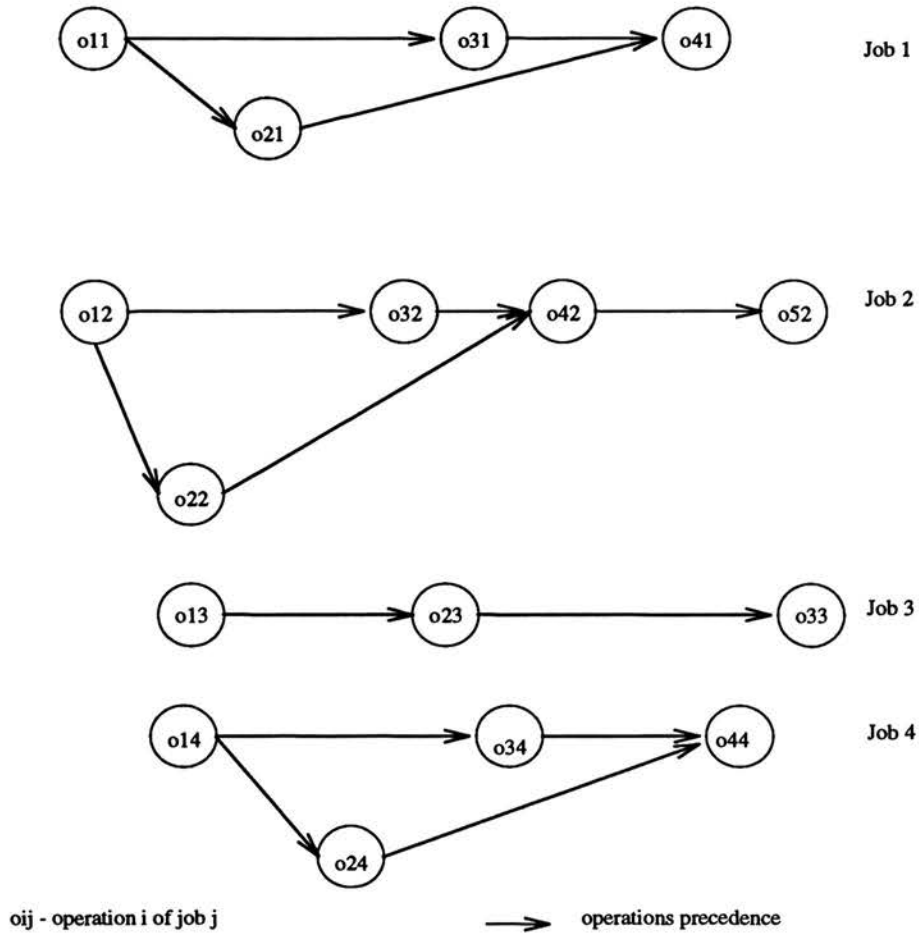

Figure 5.2: The four jobs of the scheduling problem represented in the figure 5.1

shows the four jobs represented in figure 5.1 that correspond to subgraphs (O_1, A_1), (O_2, A_2), (O_3, A_3) and (O_4, A_4).

The set of disjunctive arcs $\mathcal{E}$ decomposes the graph $\mathcal{G}$ into subgraphs (O^k, R_k, E_k), where $O^k = \{o_{ij} : (\exists\ (o_{ij}\ r_{lk})), i \in N_j, j \in N, l \in M_k, k \in M\}$[2], $O = \bigcup(O^k : k \in M)$, $R_K = \{r_{ik} : i \in M_k, k \in M\}$, $R = \bigcup(R_k : k \in M)$, one for each aggregate resource or machine type, $\mathcal{M}$ the set of indices of the aggregate resources. The subgraph (O^k, R_k, E_k) corresponds to the problem associated with the aggregate resource $k \in M$, i.e., the set of individual machines of a certain type, and the operations that have to be scheduled on it. Figure 5.3 shows the four aggregate resources represented in the

[2]Note that O^k denotes the set of operations that are assigned to the same aggregate resource k, while O_k is the set of operations that belong to the same job k.

figure 5.1 and the subgraphs associated with them, i.e., the subgraphs (O^1, R_1, E_1), (O^2, R_2, E_2), (O^3, R_3, E_3) and (O^4, R_4, E_4).

The *job shop scheduling or machine sequencing problem* can be defined as follows:

> Times (start and finish times) and individual machines have to be assigned to each operation of a set of jobs, satisfying a set of constraints and considering a certain objective.

Referring to the figure 5.1, the *job shop scheduling problem* can be stated as:

> How to partition the node set $\mathcal{O}$ into subsets such that operations that are members of the same subset are assigned to the same individual machine r_{ij}, with a given start time and finish time, satisfying all the constraints and considering a certain objective (typically the minimisation or maximisation of a certain function).

The approach adopted in EXPLICIT is "divide and conquer", i.e., the decomposition of the whole problem into smaller and more manageable problems in order to reduce the overall computational complexity of the scheduling problem. Different agents are assigned different (sub)problems. Each Job Tactical Agent is responsible for assigning time windows to the operations of its job. The Job Tactical Agent responsible for job j is denoted by JTA_j. The problem associated with JTA_j is represented by the subgraph (O_j, A_j). Each Resource Tactical Agent is responsible for assigning start times and individual machines to the operations to be performed on its aggregate resource. The Resource Tactical Agent responsible for the aggregate resource k is denoted by RTA_k. The problem associated with RTA_k is represented by the subgraph subgraph (O^k, R_k, E_k). The Strategic Agent is responsible for the whole problem, in particular for coordinating the scheduling tasks of the Tactical and Operational Agents.

In the next sections, the (sub)problems assigned to the different agents of the system are presented. In order to make the explanation easier, the description of the different scheduling tasks performed by each agent follows the flow of the scheduling

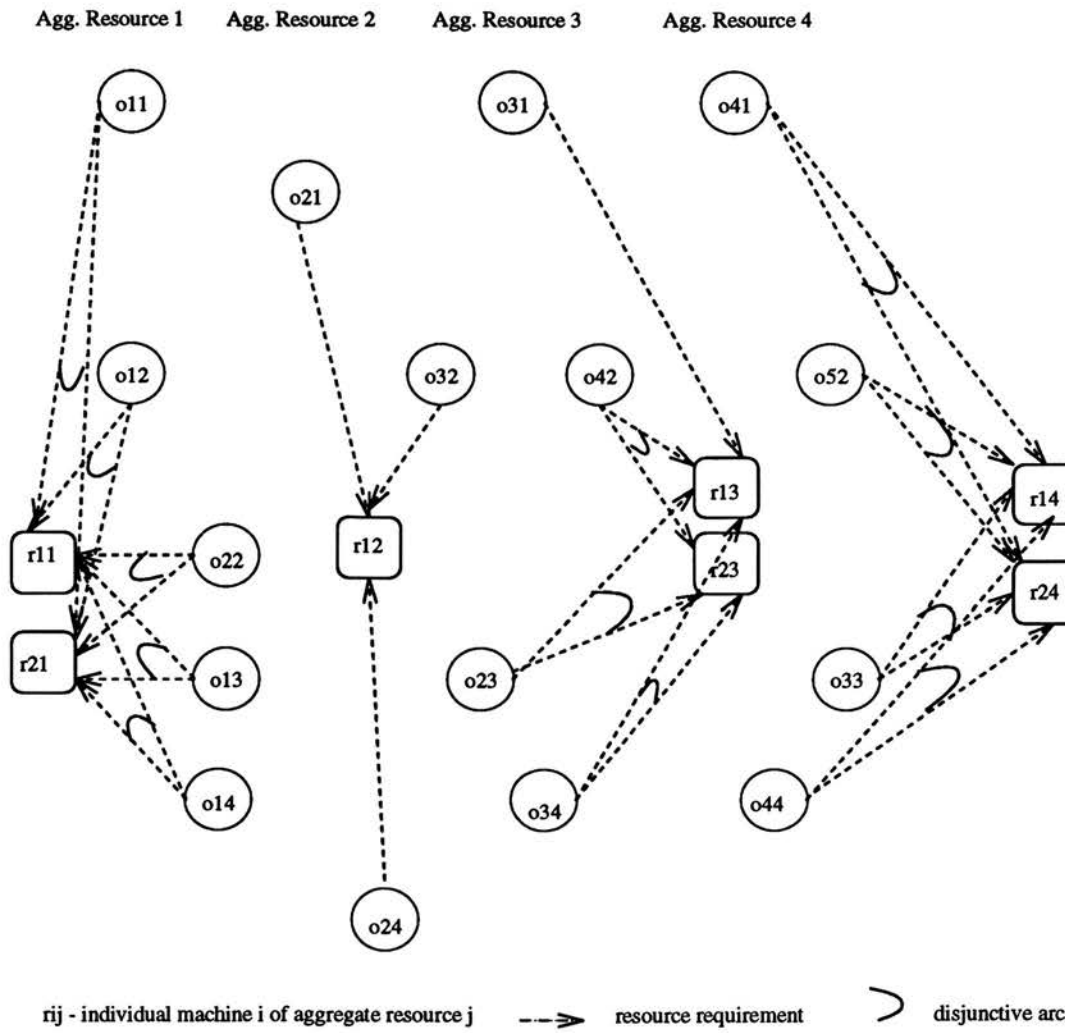

Figure 5.3: The four aggregate resources (and their operations) of the scheduling problem represented in the figure 5.1

process. Chapter 7 illustrates the scheduling process through a simple but comprehensively described example. We suggest that the reader complements the reading of the description of the algorithms presented in this chapter with the reading of the corresponding steps of the example described in chapter 7. The structure of chapter 7 closely follows the structure adopted for the description of the scheduling process in this chapter.

5.3 The Scheduling Process

The explanation of the scheduling process is decomposed into iterations, each iteration consisting of the following steps:

- Strategic Agent (SA)
  - Analysis of the Problem (Detection of Conflicts) - the SA analyses the current status of the scheduling process in order to detect conflicts. The SA sends the conflicts detected to the Job Tactical Agents (JTAs) involved in the conflicts for conflict propagation. When initialising the scheduling process (first iteration), the SA sends all the operations to the respective JTAs for time windows assignment.
- Job Tactical Agents (JTAs)
  - Assignment of Time Windows - the JTAs assign time windows to the operations of their jobs. This assignment is either a new assignment (first iteration) or a redefinition of existing time windows due to conflict propagation.
- Strategic Agent (SA)
  - The Strategic Agent's Plan - the SA defines a plan for (re)assignment of operations to the RTAs for them to assign times and individual machines to the operations. When initialising the scheduling process (first iteration), this task is trivial - all the RTAs are included in the plan for assigning times and individual machines to their operations.

- Conflict Resolution - based on its plan, the SA sends the information to the selected RTAs for (re)scheduling.

- Resource Tactical Agents (RTAs)

  - Assignment of Start Times and Individual Machines to Operations - RTAs assign individual machines and times to the operations sent by the SA. If the system includes Operational Agents (OAs), each OA optimises the times of the operations assigned to it by the respective RTA.

5.3.1 Analysis of the Problem (Detection of Conflicts)

The Strategic Agent

The SA analyses the current status of the scheduling process in order to detect conflicts. At the first iteration, all the jobs are in conflict since operations do not have time windows assigned to them. The SA sends all the operations to the respective JTAs for time windows assignment. The Strategic Agent passes all the necessary information to each JTA for time windows assignment (see also section 5.5.1).

For each job j, the SA sends to the corresponding JTA, $JTA(j)$, the information relative to the subgraph (O_j, A_j), where $O_j = \{o_{ij} : i \in N_j, j \in N\}$, $\mathcal{O} = \bigcup(O_j : j \in N)$ and $A_j = \{(o_{ij}o_{lj}) : i, l \in N_j, j \in N\}$,$\mathcal{A} = \bigcup(A_j : j \in N)$, i.e,

- $avtime(j)$ - the available time of job j
- $duedate(j)$ - the due date of job j

For each operation o_{ij} of job j, which we call simply o unless confusion might occur, the following information is sent:

- $job(o)$ - the name of the job to which the operation belongs ($JTA(j)$)
- $name(o)$ - the name of the operation
- $aggres(o)$ - the name of the aggregate resource on which the operation has to be scheduled

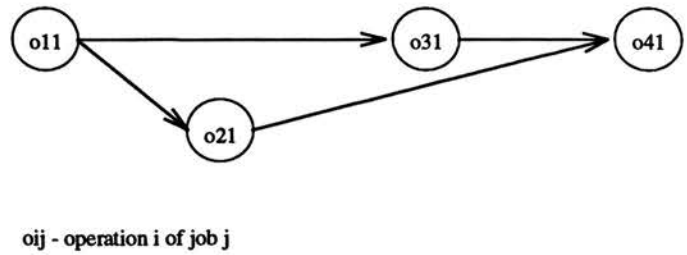

Figure 5.4: The Job Tactical Agent's problem

- *defaultduration*(o) - the default duration of the operation, independently of the particular machine where it is performed
- *suitableindmachines*(o) - the list of the individual machines where the operation can be performed
- *tableduration*(o) - the table of durations of the operation for each of the individual machines
- *after*(o) - the list of operations of the same job that are immediate successors of the operation o
- *before*(o) - the list of operations of the same job that are immediate predecessors of the operation o

5.3.2 Assignment of Time Windows

Job Tactical Agents

Each Job Tactical Agent (JTA) is responsible for assigning time windows to the operations of its job, considering the temporal constraints between operations of the same job and assuming unlimited resources. In terms of the graph $\mathcal{G}$, the Job Tactical Agent associated with job j (JTA_j) is responsible for the subgraph (O_j, A_j). Figure 5.4 illustrates the problem assigned to JTA_1, assuming the whole problem given in figure 5.1.

Each JTA assigns time windows to the operations of its job performing a critical path analysis, and independently of the other JTAs. The time windows assigned to each

operation consist of:

- $est(o)$ - earliest start time of operation
- $eft(o)$ - earliest finish time of operation
- $lst(o)$ - latest start time of operation
- $lft(o)$ - latest finish time of operation
- $slack(o)$ - slack of operation, i.e., the amount of time that the operation can be delayed without delaying the job.

The assignment of time windows is performed using a critical path method (PERT/CPM) (see e.g., [Willis 85], [Lockyer 84], [Davis & Patterson 75] and [Fendley 68]) and assuming unlimited resources. The duration considered for each operation is the default duration.

Each JTA passes on the time windows of its operations to the Strategic Agent.

5.3.3 The Strategic Agent's Assignment of Work Plan

The Strategic Agent

Based on the time windows sent by the JTAs, the SA defines a plan for assignment of work to the RTAs. At this stage of the scheduling process this task is trivial. All the operations, with the respective time windows, are sent to the respective RTA for start times and individual machines assignment (see also section 5.5.3).

The information that the SA sends to each RTA is :

- For each operation o to be scheduled on the RTA[3]:
  - $name(o)$ - the name of the operation
  - $suitableindmachines(o)$ - the list of the individual machines where the operation can be performed

[3]Note that each operation has an attribute that identifies the aggregate resource ($aggres(o)$).

- *defaultduration*(o) - the default duration of the operation
- *tableduration*(o) - the table of durations of the operation for each of the individual machine
- *est*(o) - earliest start time of operation
- *slack*(o) - slack of operation, i.e., the amount of time that the operation can be delayed without delaying the job.

5.3.4 Assignment of Times and Machines to Operations

The Resource Tactical Agents

Each Resource Tactical Agent (RTA) is responsible for assigning times and individual machines to the operations sent by the SA. RTAs delegate the operations to the corresponding Operational Agents. The Operational Agents are responsible for improving the schedule suggested by the RTA to the operations to be performed on their individual machines. If the system does not include Operational Agents, the start times and individual machines assigned by the RTAs are the ones to be sent back to the Strategic Agent.

The problem that each Resource Tactical Agent, say RTA_k, has to solve corresponds to the subgraph (O^k, R_k, E_k).

Figure 5.5 illustrates the problem assigned to RTA_3, assuming the whole problem given in figure 5.1.

The disjunctive graph $T_k = (O^k, R_k, E_k, P_k)$ represented in figure 5.6, gives another perspective of this problem.

The nodes set $\{O^k, R_k\}$ and the arcs set $\{E_k\}$ correspond to the subgraph (O^k, R_k, E_k) of the graph $\mathcal{G}$. A node that is member of the set O^k represents an operation to be performed on the resource type k (aggregate resource k). A node that is member of the set R_k represents an individual machine of type k.

The arcs set P_k, represented by full arrows in figure 5.6, does not have corresponding arcs set on graph $\mathcal{G}$. Each arc in this set represents precedence constraints for each pair

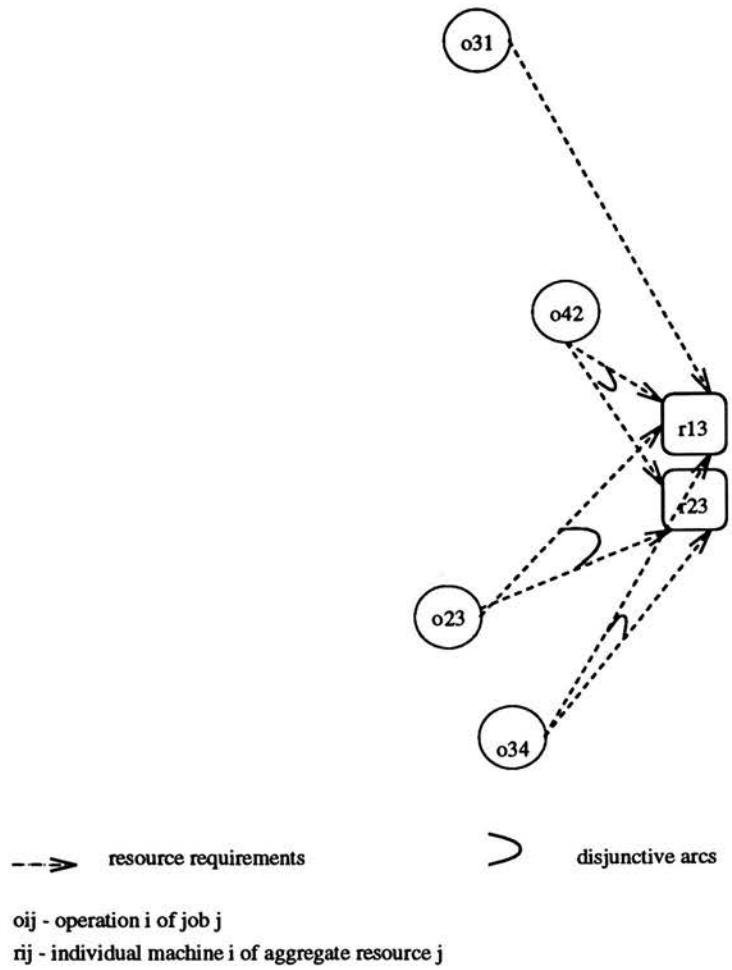

Figure 5.5: The Resource Tactical Agent's problem

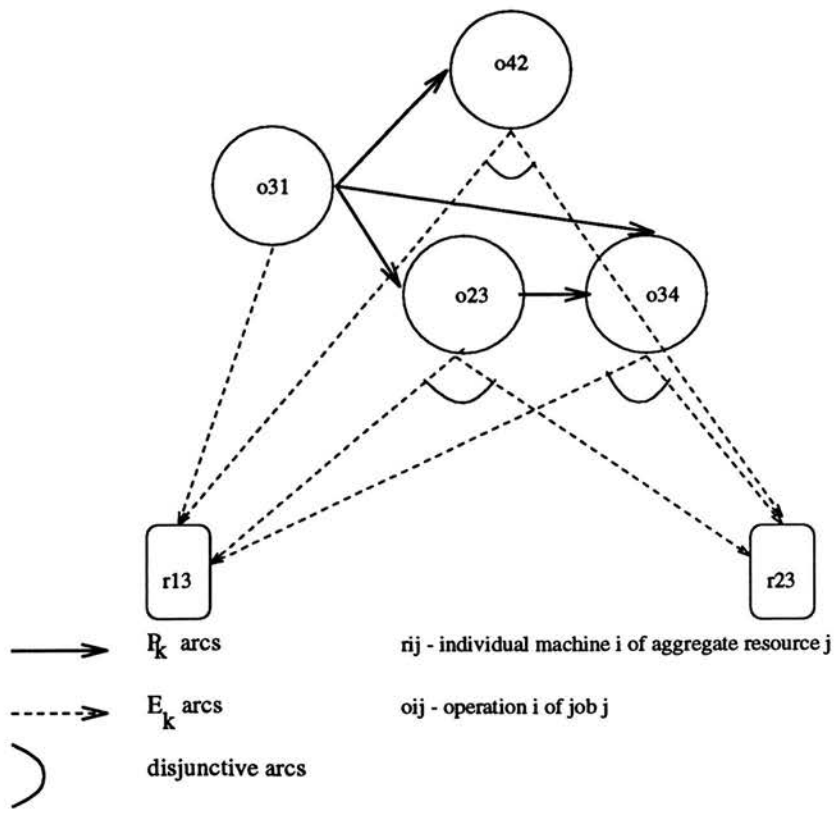

Figure 5.6: The Resource Tactical Agent's problem

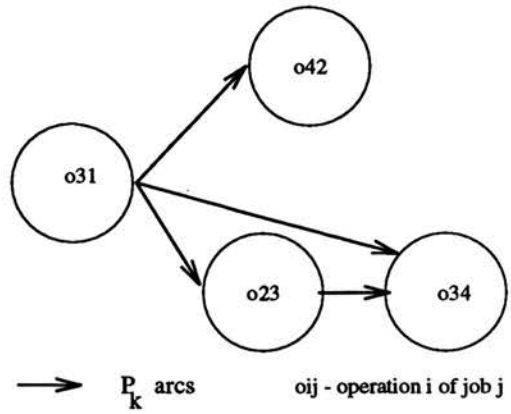

Figure 5.7: Graph I_k

of operations, in terms of the aggregate resource. Each arc of this set means that the 2 linked operations can be performed on the same resource in sequence, considering all the constraints and assuming as start time for each operation, the earliest start time of the corresponding operations. It should be noted that each arc only has a meaning for the two operations that it links, and no more. So, for instance, if there is an arc between a node I and a node J and an arc between node J and node L, one can infer that: 1) operation J can be performed after operation I on the same resource; 2) operation L can be performed after operation J on the same resource . One cannot conclude that operations I, J and L can be performed on the same resource in sequence.

The graph represented in figure 5.7 is the subgraph of graph T_k, denoted by $I_k = (O^k, P_k)$ and it is obtained by eliminating the nodes set R_k and the arcs set E_k from graph T_k.

Referring to the graph I_k, the problem assigned to each Resource Tactical Agent can be stated as:

> How to partition the graph I_k into paths such that each path corresponds to a valid sequence of operations to be performed on the same machine, each operation has a start and finish time assigned to it, and such that all the operations are assigned to individual machines?

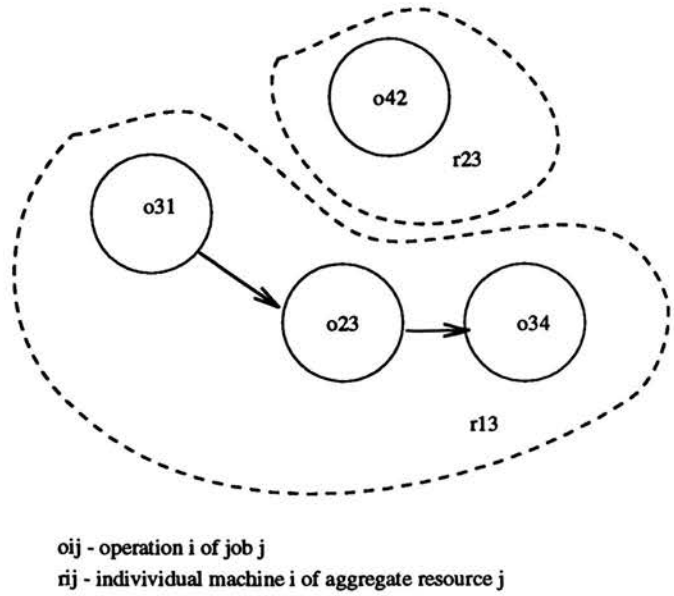

Figure 5.8: The Resource Tactical Agent's problem solution

Figure 5.8 illustrates the partitioning of the graph I_k into two paths, each one to be assigned to a different machine.

The algorithm assigned to the Resource Tactical Agents to delegate operations to the Operational agents (OAs) is described in the following paragraphs. Basically the algorithm partitions the graph I_k into paths. This algorithm is inspired by previous work in crew scheduling where partitioning approaches are very popular ([Gomes 87, Gomes & Almeida 88]), though as pointed out in chapter 2, the two types of scheduling problems have major differences (see chapter 2, section 2.3).

Assignment of Start Times and Individual Machines to Operations - Delegation of Operations to the Operational Agents

The delegation task carried out by the Resource Tactical Agents is performed by partitioning the graph I_k into paths. Figure 5.9 summarises the process followed by each Resource Tactical Agent in order to delegate work to the Operational Agents.

To perform the partitioning of the graph I_k into paths, graph I_k is modified into the graph $S_k = (S, O^k, P_k, C_k)$ (see figure 5.10). The difference between graph I_k and

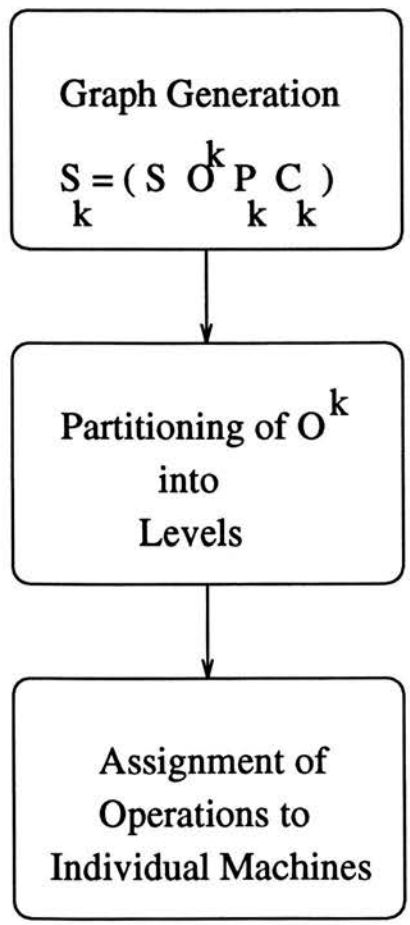

Figure 5.9: The Tactical Agent's delegation process

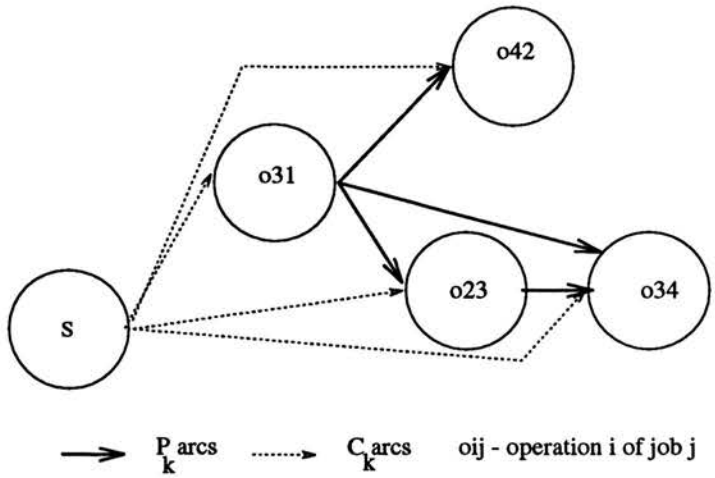

Figure 5.10: Graph S_K

graph S_k is the additional node S and the additional arcs set C_k, in graph S_k. The node S is a dummy node and it represents the aggregate set of machines, specifically each machine and the operations already assigned to it. C_k is obtained by adding one arc between each node of O^k and the node S. Each arc of C_k means that each operation will be assigned to a machine considering all the constraints (and only if all the hard constraints can be satisfied).

Generation of the graph S_k

The generation of graph S_k does not require the implicit generation of all its arcs and all its nodes. In reality the node S and the arc set C_k do not need to be generated explicitly. Also, a subset of the arcs set P_k does not need to be generated explicitly. That is the case of "transitive" arcs. If there exists an arc between node A and node B and an arc between node B and node C, an arc between node A and node C is a "transitive" arc and it does not need to be generated. The reason not to generate transitive arcs is that they do not change the complexity of the problem evaluated in terms of the numbers of levels (see definition of level below). Figure 5.10 illustrates the graph S_k and figure 5.11 illustrates the explicitly generated graph.

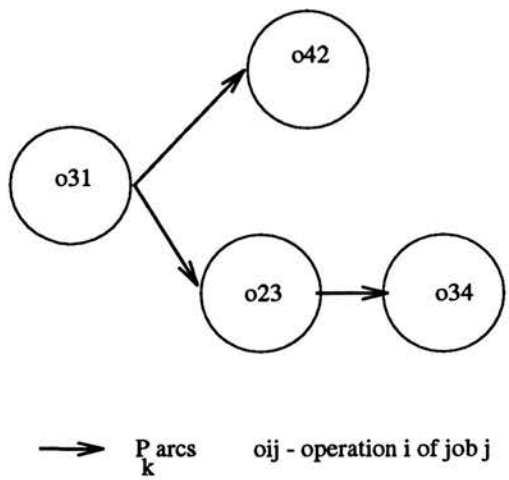

Figure 5.11: The explicitly generated Graph S_k

Partitioning of O^k into levels

The level of a node is the length of the longest path (in number of arcs) from node S to that node minus 1. Figure 5.12 illustrates the concept of level of a node.

The calculation of the level of a node is quite trivial considering that the graph S_k is a directed graph without circuits (see, e.g., [Gondran & Minoux 84]).

Assignment of start times and operations to individual machines

In order to assign times and individual machines to operations, each Resource Tactical Agent has to partition the graph S_k into paths, such that each path corresponds to a feasible sequence of operations to be performed on the same machine and every operation assigned to the Resource Tactical Agent is assigned to one (only one) individual machine. The Resource Tactical Agent performs this operation considering the times suggested to the operations by the Strategic Agent as preferential times. The schedule generated by the Resource Tactical Agent is valid from the resources point of view. Note that Resource Tactical Agents perform this scheduling process independently of each other and apart from considering the time windows of each operation as preferential constraints, they ignore other constraints related to the jobs.

The algorithm used by the Resource Tactical Agent to delegate work to the Operational Agents consists of a sequence of at least as many assignment problems (e.g., [Christofides 75] [Gondran & Minoux 84], [Minoux 84]) as the number of levels of graph S_k. Therefore it is called "Assignment Based Algorithm". Before describing the "Assignment Based Algorithm", the assignment problem, AP, is defined considering a generic Resource Tactical Agent, RTA_G, and considering that the AP is solved for a given level l of graph S_k.

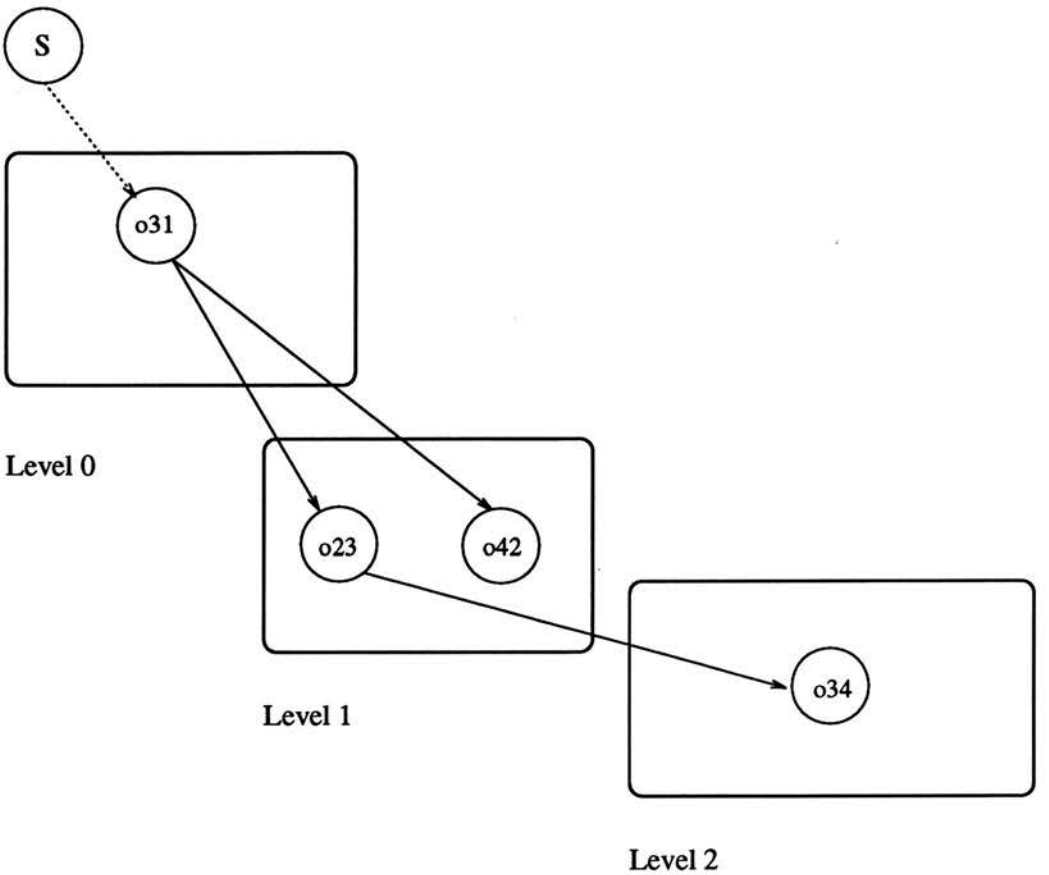

Figure 5.12: Partitioning of O^k into levels

The AP is defined as :

$$Max Z = \sum_{i=1}^{q}\sum_{j=1}^{q} u_{ij}x_{ij} \tag{5.1}$$

such that :

$$\sum_{i=1}^{q} x_{ij} = 1 \tag{5.2}$$

$$\sum_{j=1}^{q} x_{ij} = 1 \tag{5.3}$$

$$x_{ij} = 0,1 \qquad (i = 1,2,\ldots r; j = 1,2,\ldots r) \tag{5.4}$$

Where:

- RTA_G - Resource Tactical Agent responsible for the aggregate resource G.
- l - the level of the graph for which the AP is defined
- $k^{(l)}$ - the number of operations of level l that RTA_G has to assign to its machines
- p - the number of individual machines under the responsibility of RTA_G
- q - the maximum of $k^{(l)}$ and p
- op_{il} - the operation i of level l, $(i = 1,2,\ldots k^{(l)})$
- op_{il} - fictitious operation[4] i of level l, $(i = k^{(l)} + 1,\ldots q)$, if $q > k^{(l)}$
- m_j - the individual machine j, $(j = 1,2,\ldots p)$
- m_j - fictitious individual machine j, $(j = p + 1,\ldots q)$, if $q > p$
- u_{ij} - the utility associated with the assignment of op_{il} to m_j , where:
  - u_{ij} = large negative value if $i > k^{(l)}$
  - u_{ij} = large negative value if $j > p$

[4]The introduction of fictitious variables is due to the condition required by the assignment problem formulation that states that the matrix of variables has to be square. Note that the utility values associated with these variables are large negative values.

  - u_{ij} = large negative value if op_{il} cannot be assigned to m_j (violation of constraints).

- x_{ij} - decision variable, $(i = 1, 2, \ldots q; j = 1, 2, \ldots q)$

$$x_{ij} = \begin{cases} 1 & \text{if } op_{il} \text{ is assigned to } m_j \text{ with a valid start time} \\ 0 & otherwise \end{cases}$$

Equation 5.1 states that the objective is to maximise the total utility of the assignment. Equation 5.2 guarantees that one (exactly one) operation is assigned to each machine. Equation 5.3 guarantees that each operation is assigned to one (exactly one) machine. Equation 5.4 defines the decision variables.

The utility function u_{ij} adopted for each AP can be seen as a way of tuning the system to a particular domain or problem. The particular utility function embodied in the current version of EXPLICIT is presented after the description of the "Assignment Based Algorithm". However, it is important to point out that other utility functions can be defined, depending on the particular problem, domain and even on the particular objectives (see also chapter 10).

In the current implementation of EXPLICIT, the APs are solved using the shortest augmenting path algorithm (SAPA) (see e.g., [Christofides 75, Minoux 85]) for solving assignment problems. The implementation of the SAPA used to test EXPLICIT is written in C, adapted from the code described in [Balas & Saltzman 91]. The code described in [Balas & Saltzman 91] is based on FORTRAN code described in [Burkard & Derigs 80].

Assignment Based Algorithm

- For each iteration[5] N, starting with N=0
  - Construct a graph, $G_k^N = (O_N^k, R_k^N, E_k^N)$, in the following way:
    * each node of level N of graph S_k corresponds to a node of graph G_k^N of the set O_N^k. The subset O_N^k also includes nodes of the graph S_k of levels

[5] Basically each iteration corresponds to a level of graph S_K. However, due to the delay of operations, it is possible to have more iterations than the original number of levels of graph S_K. Nevertheless, sometimes it is referred that the AP is solved for a given level l of graph S_K, which might be not completely correct.

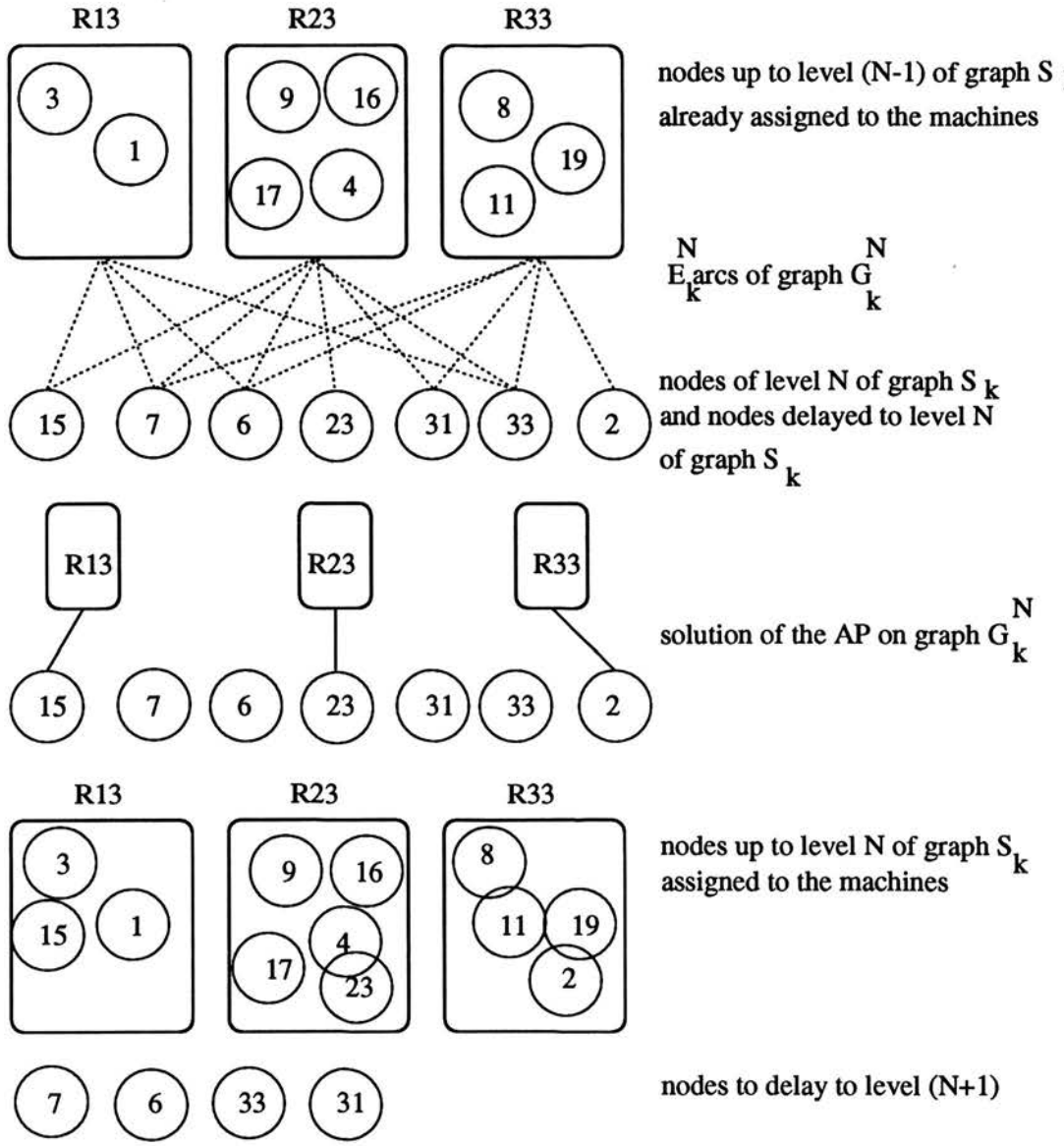

Figure 5.13: The Assignment Base Algorithm (iteration N)

lower than level N that were delayed to level N in the previous iterations of the algorithm (see the following steps of the the algorithm).

* each node of the graph G_k^N of the set R_k^N represents an individual machine, say r_{ik}, and the subset of nodes up to level (N-1) (*inc.*) of O^k of graph S_k assigned to it.
* each arc of the set E_k^N links nodes of R_k^N with nodes of the O_N^k. An arc between node R (R $\in R_k^N$) and node (O $\in O_N^k$) means that the operation represented by the node O can be assigned to the machine represented by the node R, with a given start time and finish time, considering the duration of the operation on that machine and considering all the constraints and all the previous operations already assigned to that machine. Note that an arc that is a member of E_k^N has the corresponding arc in graph $\mathcal{G} = (\mathcal{O} , \mathcal{R}, \mathcal{E})$ but the reverse is not true.

- Solve an assignment problem on graph G_k^N.
- Delay the nodes belonging to the set O_N^k that were not assigned to any machine to level (N+1). To delay a node is to shift the earliest start time of the corresponding operation to a later time. The new earliest start time of the operation is the earliest time at which one of the individual machines belonging to the aggregate resource becomes free.
- Stop condition - all the operations are assigned to a machine.

Figure 5.13 illustrates the assignment based algorithm for iteration N. After solving the set of APs every operation that was assigned to the Resource Tactical Agent has been assigned a start time, a finish time and an individual machine.

Criteria Assigned to the Resource Tactical Agents to Select the Best Assignments

Resource Tactical Agents are responsible for assigning individual machines and times to the operations to be performed on their aggregate resources. Resource Tactical Agents perform this task solving several assignment problems (APs). In order to select which

operation to assign to which machine, Resource Tactical Agents calculate the utility associated with alternative assignments and choose the assignment that maximises the total utility. The specification of the utility function associated with each of the APs is a way to tune the system for a particular domain or problem. The particular utility function that is embodied in the current version of EXPLICIT is described in this section. Other utility functions can be defined (see chapter 10).

Refer to the assignment problem (AP) that each Resource Tactical Agent has to solve for each level l of its graph, defined above.

The definition of the utility function that was adopted in the current implementation of EXPLICIT is as follows:

u_{ir} - utility associated with the assignment of op_{il} to m_r

$$u_{ir} = (1/s_{ilr} + d_{ilr} + g_{ilr}) \tag{5.5}$$

Where:

- op_{il} - the operation i of level l, $(i = 1, 2, \ldots k^{(l)})$
- $k^{(l)}$ - the number of operations of level l that the current Resource Tactical Agent has to assign to its machines
- m_r - the individual machine or resource r
- $1/s_{ilr}$ - the inverse of the relative slack of operation i of level l on resource r, where:
  -

$$s_{ilr} = \frac{\sum_{j=1,j\neq i}^{k^{(l)}} \{(slackop_{il} + gap_{ilr}) - (duration_{jlr} + gap_{jlr})\}}{\sum_{m=1}^{p} \sum_{v=1}^{k^{(l)}} \sum_{j=1,j\neq v}^{k^{(l)}} \{(slackop_{vl} + gap_{vlm}) - (duration_{jlm} + gap_{jlm})\}}$$

- $slackop_{il}$ - slack of the operation i of level l, i.e., the interval of time that the operation i of level l can be delayed without delaying the job to which it belongs;
- $duration_{ilr}$ - duration of the operation i of level l on resource r
- gap_{ilr} - gap of the operation i of level l on resource r, i.e., the interval of time between the time resource r is available to process an operation and the time operation i of level l is ready to be processed.
- p - number of individual machines
- $k^{(l)}$ - number of operations of level l

A negative value of the numerator of the fraction suggests that the operation associated to that value is critical in the sense that if it waits for other operations it risks to be delayed. The more negative value the numerator has the more critical is the operation associated with it. However, if the denominator of the fraction is also negative, the fraction becomes positive, which implies the opposite interpretation to the one just presented. To avoid this pernicious effect if the minimum value of the component corresponding to the numerator of s_{ilr} is negative ($i = 1, 2, \ldots, k^{(l)}; r = 1, 2, \ldots, p$), a scaling operation is performed transforming it into $+1$ (this value is an arbitrary positive value, ≥ 1). The same scaling is applied to the corresponding components of the other s_{ilr} ($i = 1, 2, \ldots, k^{(l)}; r = 1, 2, \ldots, p$), and consequently to the denominator of each s_{ilr}, a constant that is obtained by summing all the components that correspond to the numerators of all the s_{ilr}. After the scaling operation the most critical operation is still the one that has the smallest value of the numerator of the fraction s_{ilr}.

To illustrate this situation let us consider a simple example involving two operations, A and B. To simplify this example let us assume only one individual machine and let us also assume that both operations are ready to be processed, i.e, their *gaps* are 0. The duration and *slack* of each operation are:

- $slackop_A$ - 20

- gap_A - 0
- $duration_A$ - 20
- $slackop_B$ - 10
- gap_B - 0
- $duration_B$ - 5

	$\{(slackop_i + gap_i) - (duration_j + gap_j)\}$		
operation i	A	B	$\sum$
A	–	15	15
B	-10	–	-10

Table 5.1: The calculations of the numerator of the components s_A and $= s_B$ (before scaling)

i	*numerator of* s_i *(after scaling)*
A	26
B	1
Σ (denominator)	27

Table 5.2: The numerator of the components s_A and s_B (after scaling)

i	*numerator of* s_i *(after scaling)*	s_i	$1/s_i$
A	26	26/27	27/26
B	1	1/27	27

Table 5.3: The components $1/s_A$ and $1/s_B$

The tables 5.1, 5.2 and 5.3 illustrate the calculations and the scaling procedure in order to obtain $1/s_A$ and $1/s_B$. In this example the operation B is the most critical operation. Therefore it is the operation that has the greatest value of $1/s$ (which corresponds to the smallest value of the numerator of s).

- d_{ilr} - the relative duration of the operation i of level l on resource r where:
  –

$$d_{ilr} = \frac{duration_{ilr}}{\sum_{m=1}^{p} \sum_{j=1}^{k(l)} duration_{jlm}}$$

- $duration_{ilr}$ - duration of the operation i of level l on resource r
- p - number of individual machines
- $k^{(l)}$ - number of operations of level l

- g_{ilr} - the relative duration of the operation i of level l over the duration and the gap on resource r where :
  -

$$g_{irl} = \frac{\frac{duration_{ilr}}{duration_{ilr}+gap_{ilr}}}{\sum_{m=1}^{p}\sum_{j=1}^{k^{(l)}}\frac{duration_{jlm}}{duration_{jlm}+gap_{jlm}}}$$

  - $duration_{ilr}$ - duration of the operation i of level l on resource r
  - gap_{ilr} - gap of the operation i of level l on resource r, i.e., the interval of time between the time resource r is available to process an operation and the time operation i of level l is ready to be processed.
  - p - number of individual machines
  - $k^{(l)}$ - number of operations of level l

Equation 5.5 represents the total utility that is associated with each arc of the graph of RTA_G, for a given level l, G_G^l.

Let us examine the rationale for each component of the total utility u_{ir}:

- Component $1/s_{ilr}$ - This component is associated with the objective of meeting the due dates. It is the most important component of the utility function in terms of the selection of which operations are going to be selected for the assignment. The way this component works is by evaluating, for each operation op_{il}, the remaining slack assuming that the operation waits for the execution of the other operations on the individual machine, m_r. Operations with small remaining slack are assigned a higher utility. This component has a higher weight in the utility function and so it tends to be the decisive component. The other two components tend to be important only as tie-breakers.

- Component d_{ilr} - The underlying idea of this component is to assign a higher utility to longer operations. This component plays a secondary role in terms of the selection of which operations are going to be selected for the assignment.

- Component g_{ilr} - This component is associated with the objective of optimising the utilisation of machines. It takes into consideration the idle time induced by the assignment of a given operation to a given machine. Longer operations that involve less idle time are assigned a higher utility. This component is responsible for defining which individual machine to assign to a given operation.

In chapter 7 an example illustrates the meaning and how to calculate the different components of the utility function.

At this stage, after having solved the "Assignment Based Algorithm", all the operations have times and individual machines assigned to them. If the system is provided with Operational Agents, the Resource Tactical Agent sends the operations to the corresponding Operational Agent (OA) for the optimisation of the schedule of the operations on its individual machine. Otherwise the results are sent directly to the Strategic Agent.

The information that the RTA sends to each OA is :

- For each operation o to be scheduled on the OA:
  - $name(o)$ - the name of the operation
  - $duration(o)$ - the duration of the operation on the individual machine
  - $est(o)$ - earliest start time of operation
  - $slack(o)$ - slack of operation, i.e., the amount of time that the operation can be delayed without delaying the job.
  - $st(o)$ - start time assigned to the operation by the RTA
  - $level(o)$ - the level of the operation in graph S_k

If the system does not include OAs, the RTA sends the following information to the SA:

- For each operation o scheduled on the RTA:
  - $name(o)$ - the name of the operation
  - $duration(o)$ - the duration of the operation on the individual machine
  - $st(o)$ - start time assigned to the operation by the RTA
  - $level(o)$ - the level of the operation in graph S_k

5.3.5 Improvement of the Resource Tactical Agent's Schedules

The Operational Agents

The Operational Agents are responsible for improving the schedule of the operations sent by the Resource Tactical Agent to be performed on their individual machines.

The considerations made apropos the utility function are also applicable here. The optimisation role assigned to the Operational Agents can be seen as as a way of tuning the system to a particular domain or problem. Different algorithms can be assigned to the Operational Agents[6].

In the current implementation of EXPLICIT Operational Agents were provided with an optimal algorithm due to [McMahon & Florian 75] that minimises maximum lateness of the jobs assigned to them. For the sake of completeness, the algorithm is presented in the appendix A (see also chapter 9, section 9.1).

The information that each OA sends to the SA is :

- For each operation o scheduled on the OA:
  - $name(o)$ - the name of the operation
  - $duration(o)$ - the duration of the operation on the individual machine
  - $st(o)$ - start time assigned to the operation by the OA
  - $level(o)$ - the level of the operation in graph S_k

[6]Operational Agents of different RTAs can play different roles.

5.4 Conflict within EXPLICIT

In the previous chapter the concept of *conflict* was introduced. Furthermore, it was stated that a major thesis of the research reported here is that *conflict*, as a consequence of the scheduling process, should be exploited to allow agents to express their own interests and therefore as a way of ensuring the integration of multiple scheduling perspectives. The concept of *conflict* is fundamental within EXPLICIT. In EXPLICIT, Tactical Agents are given the opportunity to formulate their best schedules independently of each other. In this sense, exploiting *conflict* corresponds to allowing each agent to express its own interests independently of each other. Detecting and solving *conflict* corresponds to coordinating and making compatible the conflicting interests of the agents.

5.4.1 Conflict and Types of Conflicts

Conflict occurs in the following circumstances:

- when the start time assigned to an operation by the corresponding RTA does not correspond to the earliest start time assigned to that operation by the corresponding Job Tactical Agent
- when the time windows assigned to a given operation by the respective Job Tactical Agent have to be changed as a consequence of other conflicts in the same job
- when an operation has to be rescheduled because it is processed on the same aggregate resource as other operation(s) involved in conflicts and the level of that operation on that aggregate resource is equal or greater than the level of the other operation(s) involved in conflicts

These three conditions correspond to the three types of *conflicts*:

- *Conflict Generator* - this type of *conflict* occurs whenever the start time assigned to an operation by the corresponding RTA is not equal to the earliest start time

that was assigned to that operation by the corresponding JTA. The designation of *Conflict Generator* has to do with the fact that a *Conflict Generator* is a potential source of new *conflicts*. *Conflict Generators* are denoted by CG. In particular, CG_j is the *Conflict Generator* j.

- *Chain Reaction Conflict* - this type of conflict is due to the propagation of the effects of a *conflict generator* through the job where it occurred. *Chain Reaction Conflicts* are denoted by CR. In particular, CR_{ij} is the *Chain Reaction Conflict* i of *Conflict Generator* j.

- *Level Reaction Conflict* - this type of conflict is due to the propagation of the effects of a *Chain Reaction Conflict* through the operations that are processed on the same aggregate resource and that have the same or greater level[7] than the level of the operation involved in the *Chain Reaction Conflict*. The operations involved in *Level Reaction Conflicts* do not have their time windows changed but they have to be rescheduled. *Level Reaction Conflicts* are denoted by LR. In particular, LR_{il_j} is the *Level Reaction Conflict* i of the *Chain Reaction Conflict* lj.

Conflicts can have different status. The different status that a conflict can have are:

- new - the conflict was created and after its creation no other decision concerning the conflict was made;
- sent - the operation directly involved in the conflict was sent to the corresponding RTA to be rescheduled;
- cancelled - the conflict is cancelled;
- solved - the conflict is solved;

Active *conflicts*, i.e., unsolved conflicts that were not cancelled, have a status of either "new" or "sent".

Additionally, *conflicts* can originate new conflicts. This situation occurs when the RTA responsible for the operation involved in a conflict cannot reschedule that operation at

[7]The concept of level of a graph is formally defined in section 5.3.4.

the new earliest start time redefined by the SA. In this sense one can define a tree of *conflicts.*

5.4.2 The attributes of a Conflict

The following attributes define conflicts. The inclusion of the list of the attributes of *conflicts* will facilitate the understanding of the algorithms for conflict detection and conflict resolution presented in the next sections.

- number - this is a sequential number that identifies a conflict uniquely.
- job - the job directly involved in the conflict;
- resource - the aggregate resource directly involved in the conflict;
- operation - the operation directly involved in the conflict
- level - the level of the operation directly involved in the conflict in terms of the graph assigned to the corresponding RTA;
- proposed time - the last earliest time that was proposed by the corresponding JTA to the operation directly involved in the conflict;
- proposed slack - the last slack time that was proposed by the corresponding JTA to the operation directly involved in the conflict;
- start time - the start time assigned to the operation directly involved in the conflict by the corresponding RTA;
- new time - the new earliest start time of the operation directly involved in the conflict as a consequence of the conflict;
- new slack - the new slack time of the operation directly involved in the conflict as a consequence of the conflict;
- type of conflict - the type of conflict (generator; chain reaction; level reaction)
- status - the status of the conflict (new; sent; cancelled; solved)

- parentgenerator - the *Conflict Generator* at the highest level of the tree of conflicts that originated the current conflict;
- conflictgenerator - the immediate *Conflict Generator* that originated the conflict. A *Conflict Generator* is the immediate *Conflict Generator* of itself.
- parents - the list of immediate conflicts that originated the current conflict;
- children - the conflicts originated by the current conflict;

Note that *conflict* is a recursive data structure. For instance, a *conflict* might be the *conflict generator* of itself.

5.4.3 Dependencies Between Conflicts

If conflict B is dependent on conflict A (conflict A has B as a dependent), when conflict A is cancelled, B is cancelled as well. The dependency relationship is transitive. If conflict A has conflict B as a dependent and if B has C as a dependent, conflict C is also a dependent of A. Considering the "proximity" between a conflict and its dependents, one can can distinguish between "immediate dependency" (B is a immediate dependent of A) and "remote dependency" (C is a remote dependent of A). "Immediate dependency" and "remote dependency" will be referred to simply by "dependency", unless it is important to make the distinction. In ambiguous situations the term "immediate dependencies" is used to refer to a conflict and its immediate dependents.

Considering the dependencies between *conflicts*, one can define a tree of *conflicts*. Two different situations can occur:

- *Root conflicts* - a new conflict is created because the start time assigned by a given RTA to a given operation does not coincide with the first earliest start time initially assigned to that operation by the corresponding JTA. Only *conflict generators* are created in this circumstances and only some of the *conflict generators*. *Conflicts* under these circumstances are designated by *root conflicts* since they constitute the roots of the tree of conflicts.

- *Leaf conflicts* - a new conflict is created due to another conflict. *Conflicts* under these circumstances are designated by *leaf conflicts* since they constitute the internal nodes and leaves of the tree of conflicts. Three situations are possible:
  - The new conflict is a *conflict generator* that occurred as the result of the rescheduling of the operation involved in an existing conflict.
  - The new conflict is created due the propagation of the effects of an existing conflict on the job where the existing conflict occurred (*Chain Reaction Conflicts*).
  - The new conflict is created due the propagation of the effects of an existing conflict on the resource where the existing conflict occurred (*Level Reaction Conflicts*)

Default Dependencies

Default dependencies reflect dependencies that always occur between conflicts, considering the type of conflicts involved. Figure 5.14 depicts the immediate default dependencies.

Let:

- CG_k - *conflict generator k*
- CR_{ik} - *chain reaction conflict ik*
- LR_{jik} - *level reaction conflict jik*

Case 1 - conflict generator vs. its chain reaction conflicts

For a given CG_k:

$\forall i$, CR_{ik} is an immediate dependent of CG_k

Case 2 - chain reaction conflict vs. its level reaction conflicts

For a given CR_{ik}:

$\forall l$, LR_{lik} is an immediate dependent of CR_{ik}[8]

Case 3 - chain reaction conflict vs. the following chain reaction

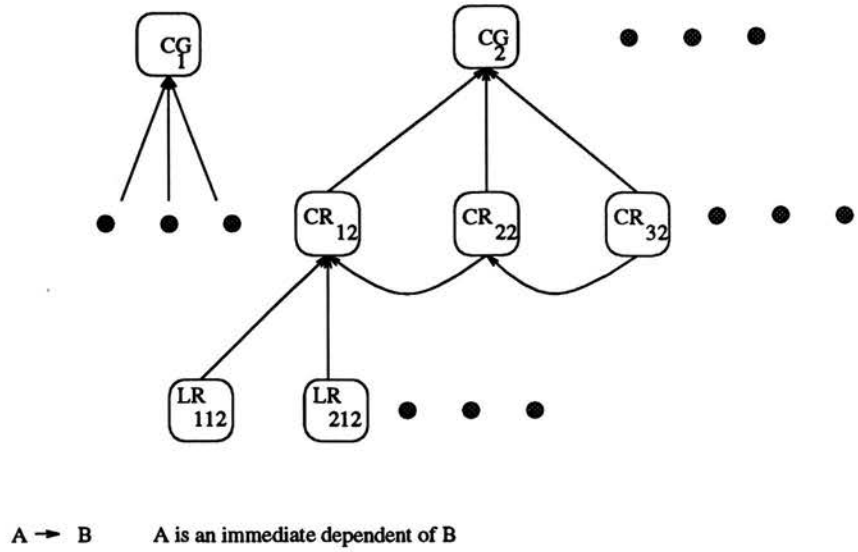

Figure 5.14: Default dependencies between conflicts

conflicts.

For a given CR_{ik}:

$\forall i$, $CR_{(i+1)k}$ is an immediate dependent of CR_{ik}[9]

The concept of default dependency is implemented through the conflict attributes of *parents*, *children*, *parentgenerator* and *conflictgenerator*.

Another type of dependency is due to the dominance of conflicts.

Dominances of Conflicts

Definition: If conflict A *dominates* conflict B, conflict B and its dependents have to be cancelled.

General Rule: a conflict with status *new* always dominates conflicts with status different from *new* involving the same operation.

[8] By the transitive rule, LR_{lik} it is also a dependent of CG_k. LR_{lik} is remotely dependent on CG_k.

[9] By the transitive rule,$\forall\, l \geq (i+1)$, CR_{lk} is dependent on CR_{ik}. $\forall\, l > (i+1)$, CR_{lk} is remotely dependent on CR_{ik}. Also, by the transitive rule, $\forall\, l \geq (i+1)$, $\forall\, m$, LR_{mlk} is dependent on CR_{ik}.

The rules of dominance of conflicts also include the following specific rules:

Let:

- c', c'' - conflicts
- CG - a *conflict generator*
- CR - a *chain reaction conflict*
- LR - a *level reaction conflict*

Rule: *Conflict generators* vs. *chain reaction conflicts*

Requirements:

$\forall(c', c'' : type(c') = (CR \text{ or } CG) \text{ and } type(c'') = (CR \text{ or } CG) \text{ and } operation(c') = operation(c''))$

Decision

IF $newtime(c') \leq newtime(c'')$
THEN c'' *dominates* c'
ELSE c' *dominates* c''

The rationale of this rule is that if two conflicts of the type *conflict generator* or *chain reaction conflict* and involving the same operation imply a new time for the operation, the conflict that dominates is the one that implies the latest time to that operation.

Rule: *Chain reaction conflicts* vs. *level reaction conflicts*

Requirements:

$\forall(c', c'' : type(c') = LR \text{ and } type(c'') = CR \text{ and } operation(c') = operation(c''))$

Decision

c'' *dominates* c'

The rationale of this rule is that if there are two conflicts involving the same operation, one of them is a *chain reaction conflict*, which means a propagation of a another conflict, the *chain reaction conflict* dominates the *level reaction conflict*. There is no point in rescheduling an operation considering its old time windows (which is the case of the *level reaction conflict*) when a priori it is known that, because of a conflict that occurred on the same job on another operation, the operation has new time windows.

Rule: *Conflict generator* vs. *level reaction conflicts*

Requirements:

$\forall(c', c'' : type(c') = LR \text{ and } type(c'') = CG \text{ and } operation(c') = operation(c''))$

Decision

c'' *dominates* c'

$type(c'') \longleftarrow CR$

This rule implies that *conflict generators* always dominate *level reaction conflicts*. Whenever a *conflict generator* and a *level reaction conflict* affect the same operation, the *level reaction conflict* is cancelled because it is dominated by the *conflict generator*, and the *conflict generator* has its type changed to CR[10] However, a rule that stated the opposite could be adopted which would mean a least commitment strategy. The rationale of using such rule is: if a set of operations to be performed on a certain aggregate resource (i.e., *level reaction conflicts*) have to be rescheduled due to a certain conflict (the *chain reaction conflict* that is parent of the *level reaction conflicts*), existing *conflict generators* involving operations on that resource that have to be rescheduled anyway should be cancelled because the new rescheduling might mean a new opportunity for the operations involved in the *conflict generators*. In other words, the original time windows of the *conflict generators* might be able to stay unchanged, due to the change of the time windows of other operations on the same resource (the

[10]This is to guarantee that the corresponding Resource Tactical Agent takes into account the operation when recalculating the level of the operations assigned to it.

ones involved in the *level reaction conflicts*). This approach involves a least commitment strategy, which means a more opportunistic approach but, on the other hand it requires more computational resources. In chapter 10 this least commitment strategy is outlined.

5.4.4 The Strategic Agent's Plan

At any time, the Strategic Agent's Plan is composed of the set of active *conflicts*, i.e., the conflicts that have status either "new" or "sent". The Strategic Agent assigns work to the Tactical Level based on its plan (see also section 5.5.3).

5.4.5 The life cycle of Conflicts

In this section the procedures related to the manipulation of conflicts are described according to the following scheme: name of the procedure, list of the parameters used by the procedure; optional parameters appear in the list of the parameters of the procedure after *&optional*; body of the procedure. A right-arrow indicates the output of the procedure. A left-arrow indicates that a value is bound to a data structure. The next section will describe the way the Strategic Agents manipulates conflicts in order to detect and solve conflicts and to re-assign work to the Tactical Level.

Creation of a conflict

The procedure for creating a conflict is described below.

Let:

- *newc* - the new conflict to be created
- *o* - the operation directly involved in the conflict
- $conflict(o)$ - conflict in which the operation *o* is involved, previously to the conflict to be created; *nil* if the operation is not involved in any conflict
- *type* - the type of the new conflict to be created

- *parents* - the parents of the conflict to be created. This case only includes the situations where the conflict to be created is either a *Chain Reaction Conflict* or *Level Reaction Conflict.*
- *conflictgenerator* - the *conflict generator* that originated the new conflict.

The procedure to create a conflict has the type of the conflict, the operation associated with the conflict and, optionally, the conflict with which the operation involved in the conflict is associated, the parents of the conflict for the case of a *chain reaction conflict* or a *level reaction conflict*[11] and the *conflict generator* of the conflict as parameters.

Procedure

$$createconflict(type\ o\ \&optional\ conflict(o)\ parents\ conflictgenerator) \longrightarrow newc$$

Body of Procedure

1 Create data structure *newc* with the following attributes:

$number(newc)$	$\longleftarrow$	sequential number	
$job(newc)$	$\longleftarrow$	$job(o)$	
$resource(newc)$	$\longleftarrow$	$aggres(o)$	
$level(newc)$	$\longleftarrow$	$level(o)$	
$proptime(newc)$	$\longleftarrow$	$est_c(o)$	
$propslack(newc)$	$\longleftarrow$	$slack_c(o)$	
$starttime(newc)$	$\longleftarrow$	$st(o)$	
$newtime(newc)$			
	$\longleftarrow$	$st(o)$	$(\ type = CG)$
	$\longleftarrow$	$newest(o)$	$(\ type = CR\ )$
	$\longleftarrow$	$est_c(o)$	$(\ type = LR)$
$newslack(newc)$			
	$\longleftarrow$	$\{slack_c(o) - (st(o) - est_c(o))\}$	$(type = CG)$
	$\longleftarrow$	$newslackop(o)$	$(type = CR)$

[11] As defined by the default dependencies

	$\longleftarrow$	$slack_c(o)$	$(type = LR)$
$type(newc)$	$\longleftarrow$	$type$	
$status(newc)$	$\longleftarrow$	new	
$parentgenerator(newc)$			
	$\longleftarrow$	$newc$	$(conflict(o) = nil$[12]$)$
	$\longleftarrow$	$parentgenerator(conflict(o))$	$(conflict(o) \neq nil)$
$conflictgenerator(newc)$			
	$\longleftarrow$	$newc$	$(type = CG)$
	$\longleftarrow$	$conflictgenerator$	$(type = CR$ or $LR)$
$operation(newc)$	$\longleftarrow$	o	
$parents(newc)$			
	$\longleftarrow$	$newc$	$(type = CG)$
	$\longleftarrow$	$parents$	$(type = CR$ or $LR)$
$children(newc)$	$\longleftarrow$	[ ]	

2 Add *newc* to the beginning of the list of conflicts affecting operation o [13]

Where:

- *newc* - the new conflict to be created
- o - the operation involved in the conflict
- $conflict(o)$ - conflict in which the operation o is involved, previously to the conflict to be created; *nil* if the operation is not involved in any conflict
- *parents* - the parents of the conflict to be created. This case only includes the situations where the conflict to be created is either a *Chain Reaction Conflict* or *Level Reaction Conflict.*

[12]This situation can only happen when the conflict to be created is a *Conflict Generator.* In this case, the conflict to be created is the parent generator of itself.

[13]Apart from the attributes listed apropos of the assignment of time windows to operations, each operation also has an attribute identifying the list of conflicts that successively affected the operation, $listconflicts(o)$. The first conflict in the list is the most recent conflict.

- *conflictgenerator* - the *conflict generator* the originated the new conflict.
- *number(newc)* - sequential number that identifies the new conflict uniquely.
- *job(newc)* - the job involved in the new conflict;
- *resource(newc)* - the aggregate resource involved in the new conflict;
- *level(newc)* - the level of the operation involved in the new conflict in terms of the graph assigned to the corresponding RTA;
- $st(o)$ - start time assigned to the operation by the corresponding RTA
- $est_c(o)$ - current existing earliest start time assigned to the operation by the corresponding JTA[14] (previous to the conflict).
- $slack_c(o)$ - current slack time assigned to the operation by the corresponding JTA[15] (previous to the conflict).
- *newest(o)* - new earliest start time assigned to the operation by the corresponding JTA due to the propagation of a conflict; this case only considers the situation where the conflict to be created is a *Chain Reaction Conflict.*
- *newslackop(o)* - new slack time assigned to the operation by the corresponding JTA due to the propagation of a conflict; this case only includes the situation where the conflict to be created is a *Chain Reaction Conflict.*
- *proptime(newc)* - the last earliest time that was proposed by the corresponding JTA to the operation involved in the new conflict;
- *propslack(newc)* - the last slack time that was proposed by the corresponding JTA to the operation involved in the new conflict;
- *starttime(newc)* - the start time assigned by the corresponding RTA to the operation involved in the new conflict;

[14]See the definition of $est_t(o)$, in section 5.5.2; in this case the moment t is c, the moment of the creation of the conflict.

[15]See the definition of $slack_c(o)$, in section 5.5.2; in this case the moment t is c, the moment of the creation of the conflict.

- *newtime*(*newc*) - the new earliest start time of the operation involved in the new conflict as a consequence of the new conflict;
- *newslack*(*newc*) - the new slack time of the operation involved in the new conflict as a consequence of the new conflict;
- *type* - the type of the new conflict
- *status*(*newc*) - the status of the new conflict;
- *parentgenerator*(*newc*) - the *Conflict Generator* at the highest level of the tree of conflicts that originated the current new conflict;
- *conflictgenerator*(*newc*) - the immediate *Conflict Generator* that originated the new conflict. A *Conflict Generator* is the immediate *Conflict Generator* of itself.
- *operation*(*newc*) - the operation involved in the new conflict
- *parents*(*newc*) - the list of immediate conflicts that originated the current new conflict;
- *children*(*newc*) - the conflicts originated by the current new conflict
- *status*(*newc*) - the status of the new conflict

Sending a conflict for rescheduling

The Strategic Agent selects conflicts among the conflicts with status "new" to be reschedule. This means that the operation involved in the conflict is sent to the respective RTA to be reschedule. In this case the *status* of the conflict is changed to "sent".

Let:

- *c* - the conflict to be sent

Procedure

sendconflict(*c*) $\longrightarrow$ *c sent*

Body of Procedure

1 Set *status* of *c* to *sent*

status(*c*) $\longleftarrow$ *sent*

A conflict that is solved

A conflict has its status changed to "solved" if, after being rescheduled[16], the start time assigned by the corresponding RTA to the operation involved in the conflict corresponds to the *newtime* proposed to the operation. Once a conflict is solved it is removed from the Strategic Agent's Plan. Additionally, the parents of the conflict that is solved lose a child, the solved conflict, and gain the children of the solved conflict, to maintain the default dependencies.

Let:

- $\mathcal{PLAN}$ - the Strategic Agent's Plan
- *c* - the conflict

Procedure

solveconflict(*c*) $\longrightarrow$ *c solved*

Body of Procedure

1 Set *status* of *c* to *solved*

status(*c*) $\longleftarrow$ *solved*

2 Remove the solved conflict from the children of its parents

[16]Rigorously speaking, the operation involved in the conflict is the one to be rescheduled.

Add the children of the conflict to the children of its parents

$\forall c' : c' \in parents(c)$:

remove c from $children(c')$

add $children(c)$ to $children(c')$

3 Remove the solved conflict from the list of the parents of its children

Add the parents of the solved conflict to the list of the parents of its children

$\forall c'' : c'' \in children(c)$:

remove c from $parents(c'')$

add $parents(c)$ to $parents(c'')$

4 Remove the solved conflict from $\mathcal{PLAN}$

remove c from $\mathcal{PLAN}$

Cancellation of a conflict

The direct reasons to cancel a conflict are:

- another conflict dominates the conflict
- the *parents* of the conflict are cancelled

Basically a conflict is cancelled when another conflict, on the same operation, dominates it. In fact, though the direct reason to cancel a conflict might be the cancellation of its parents, the indirect reason is the domination of another conflict over the conflict to be cancelled. The cancellation of a conflict implies the removal of the conflict from the children of its parents, the removal of the conflict from the Plan and the same procedure is applied to the descendants of the conflict. The procedure associated with the cancellation of a conflict is described below.

Let:

- $\mathcal{PLAN}$- the Strategic Agent's Plan
- c - the conflict to be cancelled

Procedure

$cancelconflict(c) \longrightarrow c\ cancelled$

Body of Procedure

1 Set *status* of c to *cancelled*

$status(c) \longleftarrow cancelled$

2 Remove the conflict from the children of its parents

$\forall c' : c' \in parents(c)$:
remove c from $children(c')$

3 Remove the conflict from $\mathcal{PLAN}$

remove c from $\mathcal{PLAN}$

4 Cancel the children of the conflict

$\forall c''$: $c'' \in children(c)$:
$cancelconflict(c'')$

5.5 The Scheduling Process (*cont*)

5.5.1 Analysis of the Problem (Detection of Conflicts)

The Strategic Agent

The SA is responsible for analysing the schedules generated by the RTAs to detect conflicts and to check if existing conflicts were solved or if they originated new conflicts. After having formally described the concept of *conflict* it is easier to describe the algorithm used by the Strategic Agent for detection of conflicts.

Algorithm for Detection of Conflicts

Let:

- $\mathcal{PLAN}$ - the Strategic Agent's Plan
- $\mathcal{RTA}$ - the set of aggregate resources that have sent their scheduling results to the SA[17].
- *rta* - a particular aggregate resource that belongs to $\mathcal{RTA}$
- *operations*(*rta*) - operations scheduled by the *rta*
- *o* - an operation
- $\mathcal{CG}$ - the set of *conflict generators* in $\mathcal{PLAN}$
- *cg* - a particular *conflict generator* that belongs to $\mathcal{CG}$
- $est_c(o)$ - the current earliest start time of the operation *o*[18]
- $slack_c(o)$ - the current slack time of the operation *o*[19]

[17] All the RTAs in the first iteration, one particular RTA in the other iterations.

[18] See the definition of $est_t(o)$, in section 5.5.2; in this case t is denoted by c, the moment of the creation of the conflict.

[19] See the definition of $slack_c(o)$, in section 5.5.2; in this case the moment t is c, the moment of the creation of the conflict.

- $conflict(o)$ - conflict in which the operation o is involved; nil if the operation is not involved in any conflict
- $newc$ - a new conflict to be created

For each $rta \in \mathcal{RTA}$

- For each $o \in operations(rta)$
  - If $st(o) = est_c(o)$[20]
    - If $conflict(o) \neq nil$[21]

      $solveconflict(conflict(o))$[22]
  - Else[23]
    - If $conflict(o) \neq nil$[24]

      $newc \longleftarrow createconflict(CG,\ o,\ conflict(o))$

      add $newc$ to $\mathcal{PLAN}$

      $cancelconflict(conflict(o))$
    - Else

      $newc \longleftarrow createconflict(CG,\ o)$

      add $newc$ to $\mathcal{PLAN}$

For each $cg \in \mathcal{CG}$

- If $children(cg) = nil$

  $solveconflict(cg)$[25]

At this stage the SA has added to its plan the new *conflict generators* and removed from its plan the conflicts that were solved with the (re)scheduling of the RTAs, as

[20] No conflict occurred

[21] The operation o was involved in a conflict

[22] The existing conflict was solved

[23] A new conflict generator occurred

[24] The operation o was involved in a conflict

[25] A *conflict generator* is solved when it does not have any children.

well as the conflicts that were cancelled due to the dominant conflicts.

The next step consists of the propagation of conflicts through the jobs, i.e., the generation of *chain reaction conflicts.*

The SA sends the information concerning the corresponding new *conflict generators* to each JTA, , i.e.:

- $jta(j)$ - name of the job assigned to JTA_j
- cg - a new *conflict generator*
- $\mathcal{CG}_j$ - the set of new *conflict generators* involving $jta(j)$, i.e., $\mathcal{CG}_j = \{cg : job(cg) = jta(j)\}$

5.5.2 Conflict Propagation - (Re)Assignment of time windows to operations

The propagation of conflicts throughout a certain job consists of the update of the time windows of each operation of the job, given the change of the time windows of the operation involved in the *conflict generator.* Recall that each operation is characterised by the following attributes, among others:

- $est(o)$ - earliest start time of operation
- $eft(o)$ - earliest finish time of operation
- $lst(o)$ - latest start time of operation, without delaying the job
- $lft(o)$ - latest finish time of operation, without delaying the job
- $slack(o)$ - slack of operation, i.e., the amount of time that the operation can be delayed without delaying the job.

These attributes define the time windows assigned by the respective JTA to each operation of its job, at the first assignment of time windows, the initial time windows. However operations get involved in conflicts. Recall that each operation has the attribute:

- $listconflicts(o)$ - the list of conflicts in which operation o was successively involved. The first conflict in the list is the most recent conflict affecting the operation o

Considering the initial time windows assigned to a given operation by the respective JTA and considering the history of the conflicts involving that operation, reflected in the $listconflicts(o)$, the active time windows of a given operation o at a given moment t, are:

- $est_t(o)$ - the earliest start time of operation o at a given moment t
- $eft_t(o)$ - the earliest finish start time of operation o at a given moment t
- $lst_t(o)$ - the latest start time of operation o at a given moment t
- $lft_t(o)$ - the latest finish time of operation o at a given moment t
- $slack_t(o)$ - slack of operation o at a given moment t

The active time windows of a given operation o at a given moment t are obtained according to the following rules:

- IF $listconflicts(o) = [\,]$ or $(\forall(c : c \in listconflicts(o))\ status(c) = cancelled)$
  THEN
  Active time windows for the moment $t \Leftrightarrow$ initial time windows, i.e.,
  - $est_t(o) = est(o)$
  - $eft_t(o) = eft(o)$
  - $lst_t(o) = lst(o)$
  - $lft_t(o) = lft(o)$
  - $slack_t(o) = slack(o)$
- ELSE
  Find-first $c : c \in listconflicts(o)$ and $status(c) \neq cancelled$
  Active time windows for moment t:

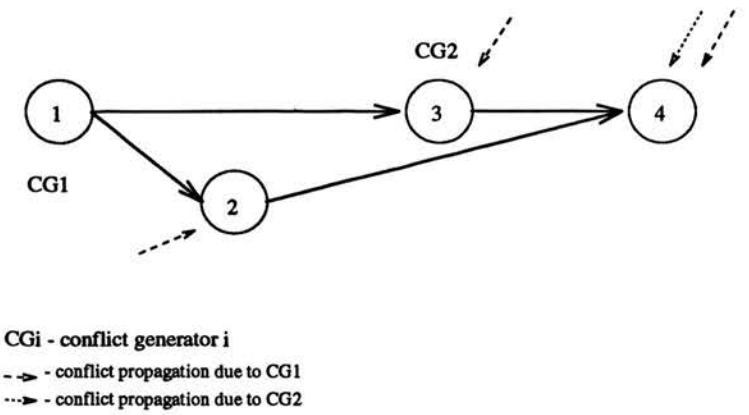

Figure 5.15: Conflicting conflicts affecting the same operation

- $est_t(o) = newtime(c)$
- $eft_t(o) = newtime(c) + defaultduration(o)$
- $lst_t(o) = lst(o)$
- $lft_t(o) = lft(o)$
- $slack_t(o) = newslack(c)$

Obviously, applying the rules just presented, the earliest start time of the operation involved in a new *conflict generator* corresponds to the *newtime* of the conflict.

The conflict propagation of a given *conflict generator* corresponds to the recalculation of the earliest start time, earliest finish time and slack of each operation that is a successor of the operation involved in the conflict, considering the current times of operations. Conflict propagation is performed using PERT/CPM techniques.

Note that several *conflict generators* can exist on the same job and, consequently, several conflicts can affect the same operation. When several conflicts affect the same operation it is considered that conflicts between conflicts exist. Figure 5.15 illustrates that situation. In this graph, the operation 3 is affected by the *conflict generator* CG2 and by the *chain reaction conflict* due to the propagation of the *conflict generator* CG1. Operation 4 is affected by the propagation of the *conflict generators* CG1 and CG2.

Each JTA sends to the SA the result of the propagation of conflicts. For each *conflict generator* sent by the SA the JTA sends the list of *chain reaction conflicts* associated with it.

5.5.3 The Strategic Agent's Assignment of Work Plan

The Strategic Agent

The Strategic Agent incorporates the new *chain reaction conflicts* sent by the JTAs into its Plan.

At this stage the Strategic Agent generates the *level reaction conflicts.* This process is simple since the Strategic Agent already holds all the information required for it. Basically, for each new *chain reaction conflict*, *cr*, several *level reaction conflicts* are created, as many as the number of operations to be performed on the same aggregate resource as the operation involved in *cr* and whose *level* is equal or greater to the *level* of the operation associated with the conflict *cr*. Recall that the operations involved in *level reaction conflicts* do not have their time windows changed. They only have to be rescheduled.

At this stage, the Strategic Agent' Plan includes all the new *conflict generators*, *chain reaction conflicts* and *level reaction conflicts* that occurred as a consequence of the assignment of the start times performed by the Resource Tactical Agents. However, conflicts between conflicts can occur, i.e., the same operation might be involved in different conflicts. In order to eliminate such conflicts between conflicts for the re-assignment of work to the Tactical Level the SA takes into consideration the rules of dominance of conflicts (see section 5.4.3).

The Strategic Agent Plan

(Removal of Conflicts Between Conflicts)

The Strategic Agent applies the rules of dominance of conflicts to its current plan. Firstly the Strategic Agent applies the rule "*Conflict generators* vs. *chain reaction conflicts*". All the dominated conflicts are cancelled from its plan. Secondly the rule

Conflict generator vs. *level reaction conflicts* is applied to all pairs of conflicts that remained in the plan, and the dominated conflicts are cancelled from the plan as well. Thirdly the rule "*Conflict generator* vs. *level reaction conflicts*" is applied and again the dominated conflicts are cancelled form the plan.

Let:

- $\mathcal{PLAN}$ - the Strategic Agent's Plan for conflict resolution, i.e., the set of active conflicts detected by the Strategic Agent

$\forall(c', c'' : c', c'' \in \mathcal{PLAN})$

- apply rule "*Conflict generators* vs. *chain reaction conflicts*";
- cancel dominated conflicts from $\mathcal{PLAN}$

$\forall(c', c'' : c', c'' \in \mathcal{PLAN})$

- apply rule *Conflict generator* vs. *level reaction conflicts*;
- cancel dominated conflicts from $\mathcal{PLAN}$

$\forall(c', c'' : c', c'' \in \mathcal{PLAN})$

-apply rule *Chain reaction conflicts* vs. *level reaction conflicts*;
-cancel dominated conflicts from $\mathcal{PLAN}$

At this stage the Strategic Agent's Plan does not have conflicting conflicts. It only has redundant *level reaction conflicts*, several *level reaction conflicts* that have the same attributes, apart from the number of conflict. For each group of identical *level reaction conflicts* the Strategic Agent removes all the conflicts, except one, making sure that the default dependencies are maintained. In order to remove redundant *level reaction conflicts* the Strategic Agent adopts the following procedure:

Let:

- $\mathcal{PLAN}$ - the Strategic Agent's Plan for conflict resolution, i.e., the set of active

conflicts detected by the Strategic Agent, after the conflicting conflicts

- $\mathcal{LR}$ - the set of *level reaction conflicts* in $\mathcal{PLAN}$
- lr - a level reaction conflict
- $\mathcal{LR}_i$ - the set of *level reaction conflicts* that involve the same operation.

$$\bigcup \mathcal{LR}_i = \mathcal{LR} \quad \bigcap \mathcal{LR}_i = \emptyset \text{ (i=1, ..., n)}$$

$\mathcal{LR}_i = \{lr_i\} \cup (\mathcal{LR}_i \setminus \{lr_i\})$, where lr_i is an arbitrary conflict of $\mathcal{LR}_i$, $\{lr_i\} \neq \emptyset$

For i=1 to n

- For each $lr \in (\mathcal{LR}_i \setminus \{lr_i\})$
  - $cancelconflict(lr)$
  - add lr_i to $children(parents(lr))$ [26]
  - add $parents(lr)$ to $parents(lr_i)$ [27]

At this stage the Strategic Agent holds the final plan for conflict resolution. The plan involves all the *conflict generators*, *chain reaction conflicts* and *level reaction conflicts* that have not been cancelled nor solved. The conflicts are sorted by ascending order of the *proptime*. In this final plan, no operation is involved in more than one conflict. The next step is to reassign work to the Resource Tactical Agents.

5.5.4 Conflict Resolution

The Strategic Agent

Based on its Plan, the Strategic Agent reassigns work to the Resource Tactical Agents.

[26] This guarantees that the parents of each *level reaction conflict* that was cancelled will keep an equivalent *level reaction conflict* as child, the conflict lr_i

[27] This guarantees that the remaining *level reaction conflict* will have the parents of the cancelled *level reaction conflicts* as its parents as well

The next Resource Tactical Agent to reschedule operations is the Resource Tactical Agent associated with the conflict that has the earliest *proptime* and whose *type* is either *chain reaction conflict* or *level reaction conflict. Conflict Generators* are solved when all their dependents are solved.

Let:

- $\mathcal{PLAN}$ - the Strategic Agent's Plan
- c - a conflict of the Plan
- c' - the conflict of the Plan that has the earliest *proptime* and $type(c') = (CR$ or $LR)$
- $resource(c)$ - the aggregate resource involved in *conflict* c
- $\mathcal{C}$ - the set of conflicts in the Plan involving the same aggregate resource as the conflict c' , with type CR or LR, i.e.:
  $\mathcal{C} = \{c : c \in \mathcal{PLAN}, resource(c) = resource(c'), type(c) = (CR\ or\ LR)\}$

The Strategic Agent sends to the RTA responsible for $resource(c')$ the set of operations involved in $\mathcal{C}$.

For each conflict $c \in \mathcal{C}$ the following information is sent to the RTA concerning the operation involved in c:

- name(operation(c)) - name of the operation
- newtime(c) - new earliest start time of the operation
- newslack(c) - new slack time of the operation

The scheduling process continues in a repetitive way. The Resource Tactical Agent, with the intervention or not of the Operational Agents, assigns times to the operations that were sent by the Strategic Agent. The results of the rescheduling are sent to the Strategic Agent. The Strategic Agents analyses the results sent by the Resource

Tactical Agent in order to redefine its Plan considering the conflicts that were solved, the new detected conflicts, due to the assignment of start times performed by the Resourtce Tactical Agents and due to the conflict propagation performed by the Job Tactical Agents, as well as the conflicts that have to be cancelled due the dominance of other conflicts. After redefining its Plan a new Resource Tactical Agent is selected to reschedule operations on its resource. The scheduling process terminates when the Strategic Agent's Plan is an empty plan, which means there are no more conflicts in the schedules proposed by the Tactical Agents, or when unsolvable conflicts are detected. In chapter 10 the possibility of having several Resource Tactical Agents (re)scheduling their tasks concurrently is discussed.

5.6 Summary

In this chapter a detailed description of the scheduling process adopted in EXPLICIT is presented. The general features of the type of problems tackled by EXPLICIT are also described in this chapter.

A Distributed Problem Solving (DPS) approach is adopted in EXPLICIT. EXPLICIT can be compared to a hierarchical organisation with three main levels: the Strategic level, the Tactical level and the Operational level. This structure is inspired by DAS ([Burke 89], [Buchanan *et al* 89], [Burke & Prosser 89]). However, although there are some similarities between EXPLICIT and DAS in terms of the general structure of the system, there are substantial differences in terms of the processes associated to the different agents of the systems, i.e., the functional organisation of the systems, and in terms of the techniques and the methods used in both systems. The scheduling process adopted in EXPLICIT consists of: analysis of the problem and detection of conflicts performed by the Strategic Agent; assignment of time windows performed by the Job Tactical Agents; assignment of start times and individual machines performed by the Resource Tactical Agents. Operational Agents are responsible for optimising the schedules suggested by the Resource Tactical Agents. The concept of conflict is a central concept in EXPLICIT. The approach adopted in EXPLICIT combines a conflict resolution algorithm, a critical path method, CPM, (see, e.g., [Willis 85], [Davis & Patterson 75]

and [Fendley 68]), an assignment based algorithm, which integrates an assignment algorithm (see, e.g., [Gondran & Minoux 84]) and an optimal algorithm for minimising maximum lateness ([McMahon & Florian 75]). EXPLICIT performs a sequence of optimisations of the load balancing of the resources, each time assigning times and resources to the operations considered more critical. Each operation is analysed individually (*operation based perspective*) and its criticality results from the combination of a *job based perspective* (*chain reaction conflicts*) with a *resource based perspective* (*level reaction conflicts* and *conflict generators*).

Chapter 6

The Organisational Interpretation of EXPLICIT

In this chapter EXPLICIT is analysed from an organisational point of view, i.e., in terms of roles and organisational functions assigned to the different agents of the system, its communication and information flows and its control regime.

6.1 The organisational Structure of EXPLICIT

In chapter 3 it was pointed out that looking at a system from a distributed problem solving perspective enforces the explicit consideration of aspects such as decomposition of the problem into smaller problems, agents modelling in terms of their skills and interaction of agents with each other. A distributed system is a collection of agents and can be viewed as an organisation [Malone & Smith 84]. Viewing a distributed system as an organisation helps to understand and to define the structure of the system: processes, communication paths and control regime. The selection of an organisational structure for a system enforces the explicit definition of the roles assigned to the different agents (functional, products or markets or other) as well as the relationships of authority, communication control and information flow [Gomes & Beck 92].

The organisational structure of EXPLICIT can be classified as an uniform hierarchy. Basically, in a uniform hierarchy there are multiple levels of management to ensure proper and decentralised decisionmaking [Gomes & Beck 92]. Each level of the hierar-

chy acts as a filter of the information and the decisions that are propagated through the hierarchy. Different functions, responsibilities and decision capabilities are assigned to the different agents of the system depending on their level in the hierarchy. Top level agents have more global responsibilities while lower levels perform more local decisions.

In EXPLICIT, the Strategic Agent is responsible for the whole problem, particularly for assigning work to the Tactical Level and for solving conflicts that occur as a consequence of the asynchronous scheduling decisions performed by the Tactical Agents. Tactical Agents have special skills and specific authority to solve their scheduling problems. The Resource Tactical Agents delegate work to the Operational Agents. At the lowest level of the hierarchy, the Operational Agents are responsible for locally improving the schedule suggested by the Resource Tactical Agents.

In the next section the algorithms assigned to the different agents of EXPLICIT are analysed from a functional point of view.

6.2 The Functions and Roles assigned to the Agents

As mentioned in chapter 3, it is important to provide a scheduler with different functional perspectives and particularly with capabilities of analysis, goal definition & task planning and evaluation functions. In this section the scheduling tasks performed by each agent are analysed from a functional point of view.

6.2.1 The Strategic Agent

Analysis Capabilities

The Strategic Agent is responsible for the analysis of the whole problem in order to to delegate work to the Tactical Level and to solve conflicts that occur as a consequence of the scheduling decisions performed by the Tactical Agents. The set of analysis mechanisms assigned to the Strategic Agent are tools to ensure that the Strategic Agent is a *structurally aware* agent, capable of looking at the problem from a global perspective. In particular, the Strategic Agent is provided with:

- Conflict analysis tools to analyse the conflicts that might occur as a consequence of the asynchronous decisions performed by the Tactical Agents (Job Tactical Agents and Resource Tactical Agents) when defining the best schedule for their operations, from a job and resource perspective respectively. The conflict analysis mechanisms provide the Strategic Agent with information to (re)define goals and plans to (re)assign tasks to its subordinate Tactical Agents. The most important conflict analysis mechanisms provided to the Strategic Agent are:
  - The categorisation of conflicts into types with the corresponding properties (taxonomy of conflicts)
  - The definition of dependencies of conflicts
  - The definition of criticality and dominance of conflicts
- Goal definition and task planning tools to analyse the schedules produced by the Tactical Agents form a global perspective, considering the criticality of each operation and considering the complexity of the rescheduling task. These tools include in particular:
  - The concept of structuring a problem into levels developed by the Tactical Agents which provide information to evaluate the complexity of the rescheduling process in terms of:
    * conflict propagation on the jobs
    * conflict propagation on the resources

Planning Capabilities

The Strategic Agent elaborates plans for the assignment and re-assignment of tasks to the lower levels of the system. The Strategic Agent builts its plan taking into consideration the conflicts that arise from the asynchronous scheduling processes performed by the Tactical Agents.

Evaluation Capabilities

The Strategic Agent is assigned with mechanisms to evaluate and criticise the several

schedules generated by the Tactical Agents. In particular, the Strategic Agent is provided with mechanisms to criticise the solution from a global point of view (all jobs, all resources) and considering the criticality of individual operations. The reassignment of tasks performed by the Strategic Agent takes into consideration the criticality of each operation. The criticality of each operation (*operation based perspective*) results from the combination of a *job based perspective* (time windows and *chain reaction conflicts*) with a *resource based perspective* (*conflict generators* and *level reaction conflicts*).

Execution Capabilities

The Strategic Agent is responsible for:

- Identifying the conflicts that exist among the schedules proposed by the different Tactical Agents.
- Generating Plans to solve the detected conflicts
- Generating Plans to coordinate the scheduling activity of the Tactical Agents

6.2.2 The Job Tactical Agents

Analysis Capabilities

The Job Tactical Agents are also provided with mechanisms to ensure its *structural awareness.* The Job Tactical Agents are provided with a critical path analysis method to analyse the requirements of each job in order to define the subgoals in terms of operations (time windows). Inherent to the PERT/CPM techniques, the Job Tactical Agents decompose their problems into levels, which constitutes a tool to analyse the opportunities and conflicts to perform the time windows assignment and to perform the propagation of conflicts on their jobs.

Goal Definition & Task Planning

The critical path method provided to the Job Tactical Agents constitutes their own planning tools for time windows assignment and conflict propagation.

The Job Tactical Agents do not delegate work to other agents.

Evaluation Capabilities

The Job Tactical Agents evaluate the criticality of the operations of their jobs based on the different types of "slacks".

Execution Capabilities

The Job Tactical Agents are responsible for:

- The assignment of time windows to the operations of their jobs
- The propagation of conflicts throughout their jobs

6.2.3 The Resource Tactical Agents

Analysis Capabilities

The most powerful mechanism assigned to the Resource Tactical Agents to analyse the complexity of the subproblems that are assigned to them, i.e., to make Resource Tactical Agents *structurally aware* of their problems, is the level decomposition tool. The partitioning of the whole (sub)problem into levels gives the Resource Tactical Agents a notion of the opportunities and conflicts that are associated with the different jobs to be scheduled on its resource. The concept of level provides the Resource Tactical Agents with a mechanism to assign priorities to each job considering the opportunities and conflicts detected. Another analysis mechanism assigned to the Resource Tactical Agents is the ability for them to represent their problems through graphs (T_K, S_K, and for each level N, G_K^N), whose links denote opportunities in terms of (re)assignment of operations to the machines.

Goal Definition & Task Planning

A major mechanism assigned to the Resource Tactical Agents to define goals and to define a plan to assign operations to the lower level (their Operational Agents) is the assignment based algorithm which includes the assignment algorithm(see, e.g., [Gondran & Minoux 84]).

Evaluation Capabilities

The Resource Tactical Agents are provided with an utility function to criticise alternative assignments of start times and machines to the operations to be performed on their resources. The utility function is a way to tune the system for a particular domain and problem. In chapter 5 a particular utility function is presented. In chapter 7 the application of the particular utility function presented in chapter 5 is described.

Execution Capabilities

The Resource Tactical Agents are responsible for:

- Assigning times to the operations to be scheduled on its resources
- Elaborating a Plan to coordinate the scheduling activity of the Operational Agents

6.2.4 The Operational Agents

In the current implementation of EXPLICIT Operational Agents are assigned an algorithm to minimise the maximum lateness [McMahon & Florian 75]. The considerations in this section refer to the characteristics of that algorithm (see appendix A for a description of the algorithm).

Analysis Capabilities

The Operational Agents are provided with mechanisms to analyse the schedule of the operations that are sent by the corresponding Resource Tactical Agent to locally improve it. In particular, the concept of critical job and generating set provides an indication on how to reschedule operations.

Goal Definition & Task Planning

The branch and bound algorithm assigned to the Operational Agents constitutes the plan for rescheduling operations on their individual machines.

The Operational Agents do not delegate any work to other agents.

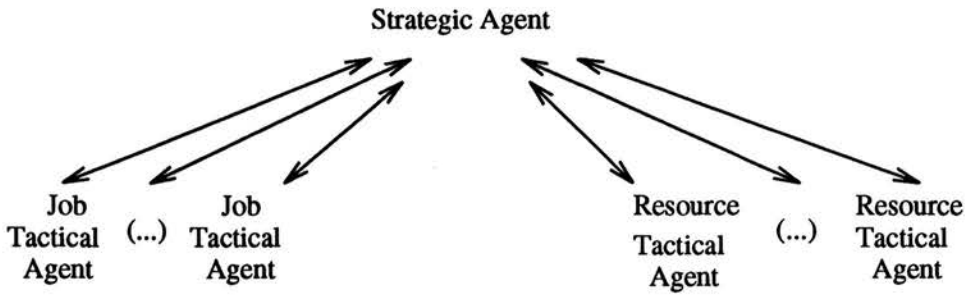

Figure 6.1: The communication and information flows - system without Operational Agents

Evaluation Capabilities

The Operational Agents are provided with evaluation criteria for the improvement of the schedule of the operations on their individual machines. The bounds associated with the branch and bound algorithm provide an indication on when an optimal solution is reached.

Execution Capabilities

The Operational Agents are responsible for locally improving the schedule suggested by the Resource Tactical Agent.

6.3 Communication & Information Flows and Control Regime

Figures 6.1 and 6.2 depict the communication and information flows among agents, for the case where the system includes Operational Agents or not. The communication and information flows are associated with the three major cycles of interaction among the agents of the system:

- The interaction between the Strategic Agent and the Job Tactical Agents for time windows assignment;
- The interaction between the Strategic Agent and the Resource Tactical Agents for start times and individual machines assignment;

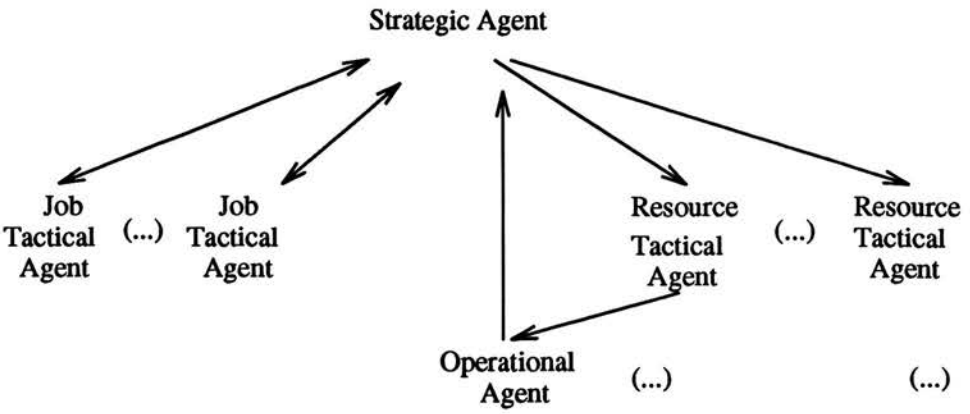

Figure 6.2: The communication and information flows - system including Operational Agents

- The interaction between the Resource Tactical Agents and the Operational Agents for optimisation of the schedules of the individual machines.

In EXPLICIT, all the communication among agents flows vertically. There is no horizontal communication among agents.

In the previous chapter the detailed description of the data interchanged among agents was presented. Different information is assigned to the different agents of the system, depending on their level in the hierarchy. Top level agents have more global information while lower levels have more local information. In general terms, the information held by the different agents of the system is:

- Strategic Agent holds information concerning the time windows and the start times assigned to the operations, measures of the complexity of the scheduling problems assigned to the Tactical Agents, the levels of the operations from the Tactical Agents point of view, and information concerning its own scheduling process, in particular the conflicts and the plans for conflict resolution.
- Job Tactical Agents hold information concerning the operations of their jobs, including the conflicts involving those operations, and their own scheduling processes.

- Resource Tactical Agents hold information concerning their aggregate resources, i.e., the set of individual machines capable of performing a given operation, the time windows of the operations to be scheduled on them and the information concerning their scheduling processes.

- Operational Agents hold information concerning their individual resources, the time windows of the operations to be scheduled on them and the information concerning their scheduling processes.

In terms of control, the Strategic Agent has the global control over the scheduling process. The Strategic Agent delegates work to the Tactical Agents. The Strategic Agent defines when and which scheduling tasks each agent should perform. Furthermore, the Strategic Agent is responsible for accepting or rejecting the scheduling decisions performed by the Tactical Agents and also, according to its plans, the Strategic Agent defines how conflicts should be solved. Resource Tactical Agents also have delegating functions. They control the scheduling process in terms of resources. They delegate work to the Operational Agents

6.4 Summary

In this chapter EXPLICIT was analysed from the organisational point of view, i.e., in terms of the functions and roles assigned to each agent, the communication flows among agents and its control regime.

Chapter 7

How EXPLICIT Works: The Newspaper Example

The main purpose of this chapter is to illustrate how EXPLICIT works through a simple but comprehensively described example. The example used takes inspiration from [French 82].

7.1 The Newspaper Reading Problem

Alan, Carla, Flavio, Ian, Nelson and Suresh share a flat. Every Saturday they have delivered two copies of the following newspapers: the European (eur or e), the Financial Times (fin or f), the Guardian (gua or g), the Scotsman (sco or s). Each member of the flat gets up at a certain time and insists on reading all the papers in a particular order (precedence constraints). Each member of the flat wants to leave the flat by a given time (due-time). Table 7.1 summarises the data for the example[1].

In this example, each reader represents a job. The available time for each job corresponds to the time the reader gets up. The due-time of a job corresponds to the latest time the reader wants to leave the flat. In this example, operations of each job correspond to the act of reading a newspaper by a reader. The precedence constraints, i.e., the order that each reader wants to read the newspaper, are reflected in the order of the columns in table 7.1. Table 7.2 represents the availability of newspapers in

[1]Time is converted into sequential minutes (e.g., 525 = 8 hours * 60 minutes + 45 minutes).

terms of time and number of copies. Newspapers correspond to resources in a job-shop scheduling problem. Each copy of a particular newspaper corresponds to an individual machine.

Reader	*Gets* Up At	*Due* Time	*1st Paper*		*2nd Paper*		*3rd Paper*		*4th Paper*	
				Time		Time		Time		Time
Alan	510	630	F	60	G	30	E	2	S	5
Carla	525	660	E	5	G	15	F	10	S	30
Flavio	585	690	G	5	S	5	E	5	F	60
Ian	570	690	S	90	F	1	G	1	E	1
Nelson	540	660	F	30	S	10	E	4	G	10
Suresh	525	640	G	75	E	3	F	25	S	10

Table 7.1: The data for the newspaper reading problem

Newspaper	*Number of Copies*	*Delivered Time*
fin	2	510
eur	2	520
gua	2	510
sco	2	525

Table 7.2: The availability of the newspapers

7.2 The Agents of the System

The agents for the newspaper problem are:

- The Strategic Agent (SA) - responsible for the whole scheduling problem.
- The Job Tactical Agents (JTAs) - one Job Tactical Agent per job (reader): Alan, Carla, Flavio, Ian, Nelson and Suresh.
- The Resource Tactical Agents (RTAs) - one Resource Tactical Agent per type of resource (newspaper): the European (eur), the Financial Times (fin), the Guardian (gua) and the Scotsman (sco).
- The Operational Agents (OAs) - one Operational Agent per individual machine (copy of a newspaper): copy number 1 of the European (eur1), copy number 2 of

the European (eur2), copy number 1 of the Financial Times (fin1), copy number 2 of the Financial Times (fin2), copy number 1 of the Guardian (gua1), copy number 2 of the Guardian (gua2), copy number 1 of the Scotsman (sco1), copy number 2 of the Scotsman (sco2)

7.3 The Scheduling Process

The description of the scheduling process focuses on the interaction between agents and on the role of the Strategic Agent in terms of conflict detection and conflict resolution. The algorithms assigned to each agent (SA, JTAs, RTAs and OAs) in order to perform its task are not discussed in detail in this section (for a more detailed description of the algorithms see chapter 5). To facilitate the explanation of the scheduling process, it will be decomposed into iterations, as defined in chapter 5, each iteration consisting of all or some of the following steps:[2]

- SA
  - Analysis of the Problem (Detection of Conflicts) - the SA detects conflicts that occurred as result of the assignment of start times by the RTAs. The SA sends the detected conflicts to the JTAs involved in the conflicts. When initialising the scheduling process, the SA sends all the operations to the respective JTAs for time windows assignment.
- JTAs
  - Assignment of Time Windows - the JTAs assign time windows to the operations of their jobs. This assignment is either a new assignment or a redefinition of existing time windows due to conflict propagation.
- SA
  - The Strategic Agent's Plan - the SA defines a plan for (re)assignment of start times to the operations for the RTAs.

[2]In the description of the scheduling process, each step will be labeled with the respective iteration number. For instance, detection of conflicts (1) refers to the step "detection of conflicts" of the first iteration of the scheduling process.

- Conflict Resolution - based on its plan, the SA sends information to the selected RTAs for (re)scheduling.

- RTAs

  - Assignment of Start Times to Operations - the RTAs assign machines and times to the operations sent by the SA. Each OA optimises the times of the operations assigned to it by the respective RTA.

7.3.1 Iteration 1

Analysis of the Problem (Detection of Conflicts) (1)

At the first iteration, all the jobs are in conflict since operations don't have time windows assigned to them. The SA initiates the scheduling process by sending all the operations to the respective JTAs for time windows assignment. The Strategic Agent passes all the necessary information to each JTA for time windows assignment.

Assignment of Time Windows To Operations (1)

The Job Tactical Agents

Each JTA assigns time-windows to its operations solving a critical path method problem (see e.g., [Willis 85], [Lockyer 84], [Davis & Patterson 75] and [Fendley 68]) and independently of the other JTAs. At this stage the availability of resources is not considered or, in other words, resources are considered unlimited.

Figure 7.1 illustrates the calculation of the time windows for job Alan using a *potential stage graph*, more usually referred to as the PERT[3] graph. In this figure the operations (tasks according to PERT terminology) are the arcs of the graph. The length of each arc is the duration of the associated operation. The start and finish of an operation are the stages of the project (in the example job Alan) and they correspond to the nodes of the graph. The initial stage of the project corresponds to the initial stage

[3]PERT is an acronym for "project evaluation and review technique".

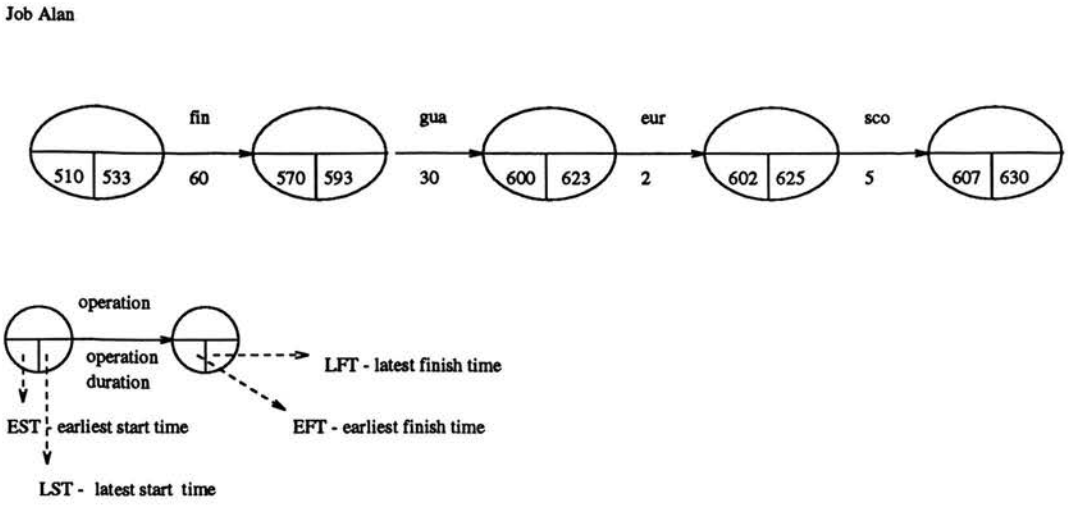

Figure 7.1: Assignment of time windows - job Alan

of the first operation. For each stage, say x_k, we define the earliest date t_k to reach that stage. It corresponds to the length of the longest path between the initial stage and stage x_k. The earliest time for the start of an operation starting from stage x_k is t_k. The earliest date of the initial stage of the project corresponds to the time job Alan is available (510). The earliest start time of operation "fin" corresponds to the earliest date of the initial stage of the project, 510. The earliest start time of operation "gua" corresponds to the earliest date to reach the start stage of operation "gua", $570 = 510 + 60$. The earliest start time of operation "eur" corresponds to the earliest date to reach the start stage of operation "eur", $600 = 570 + 30$. The earliest start time of the other operations of the job are obtained in a similar way, moving forward on the graph. The earliest finish time of an operation corresponds to the earliest date to reach the finish stage of that operation. In this example, the earliest finish time of an operation always corresponds to the earliest start time of another operation or to the finish of the job, in the case of the last operation. Analogously to the earliest date t_k to reach stage x_k we can define the latest date of the stage x_k, l_k. It is obtained by subtracting the length of the longest path between the final stage and stage x_k to the due date of the project. The latest time to finish an operation corresponds to the latest date of the finish stage of that operation. The due date of job Alan is 630. The latest finish time of operation "sco" is 630. The latest finish time of operation "eur" is

$625 = 630 - 5$. The latest finish time of operation "gua" is $623 = 625 - 2$. The latest finish time of the other operations of the job are obtained in a similar way, moving backwards on the graph. The latest start time of an operation corresponds to the latest date of the start stage of that operation. The slack of an operation corresponds to the difference between the earliest and latest start time of the operation. In this example the calculation of the time windows is trivial since the sequence of the operations is completely defined. More complicated situations occur when dealing with partially ordered networks. The time windows for the newspaper example are shown in tables 7.3, 7.4, 7.5, 7.6, 7.7 and 7.8.

JTA	*Operation*	*EST*	*LST*	*EFT*	*LFT*	*Slack*
alan	fin	510	533	570	593	23
alan	gua	570	593	600	623	23
alan	eur	600	623	602	625	23
alan	sco	602	625	607	630	23

Table 7.3: The time windows assigned by JTA Alan

JTA	*Operation*	*EST*	*LST*	*EFT*	*LFT*	*Slack*
carla	eur	525	600	530	605	75
carla	gua	530	605	545	620	75
carla	fin	545	620	555	630	75
carla	sco	555	630	585	660	75

Table 7.4: The time windows assigned by JTA Carla

JTA	*Operation*	*EST*	*LST*	*EFT*	*LFT*	*Slack*
flavio	gua	585	615	590	620	30
flavio	sco	590	620	595	625	30
flavio	eur	595	625	600	630	30
flavio	fin	600	630	660	690	30

Table 7.5: The time windows assigned by JTA Flavio

JTA	*Operation*	*EST*	*LST*	*EFT*	*LFT*	*Slack*
ian	sco	570	597	660	687	27
ian	fin	660	687	661	688	27
ian	gua	661	688	662	689	27
ian	eur	662	689	663	690	27

Table 7.6: The time windows assigned by JTA Ian

JTA	*Operation*	*EST*	*LST*	*EFT*	*LFT*	*Slack*
nelson	fin	540	606	570	636	66
nelson	sco	570	636	580	646	66
nelson	eur	580	646	584	650	66
nelson	gua	584	650	594	660	66

Table 7.7: The time windows assigned by JTA Nelson

JTA	*Operation*	*EST*	*LST*	*EFT*	*LFT*	*Slack*
suresh	gua	525	527	600	602	2
suresh	eur	600	602	603	605	2
suresh	fin	603	605	628	630	2
suresh	sco	628	630	638	640	2

Table 7.8: The time windows assigned by JTA Suresh

The Strategic Agent's Plan (1)

The Strategic Agent

The Strategic Agent (SA) collects all the data from the different JTAs and sends the operations' time-windows to the corresponding Resource Tactical Agents (RTAs). The first time the SA performs this operation all the operations with the respective time-windows are sent to the corresponding RTAs in order to have start times assigned to them.

Assignment of Start Times To Operations (1)

The Resource Tactical Agents

Each Resource Tactical Agent has to schedule the set of operations sent by the Strategic Agent on its aggregate resource. RTAs schedule the operations on their resources independently of each other. Each RTA decides which individual machine (Operational Agent) is going to perform which operation. It also suggest a start time for each operation[4]. However these start times are not mandatory. Operational Agents are allowed to modify them.

RTAs assign start times to the operations to be performed on their resources independently of each other. In order to assign start times and individual machines to its operations, the RTA solves the "Assignment Based Algorithm" - (1) the RTA builds a graph of its operations and partitions it into levels; (2) for each level, the RTA solves an assignment problem to assign machines (and start times) to each operation of that level. Operations that cannot be assigned to a machine are delayed to the next level. After solving the "Assignment Based Algorithm", operations have start times and individual machines assigned to them. At this stage the RTA sends the operations to the

[4]As pointed out before, the system can work without Operational Agents. In that case, the start times assigned by the Resource Tactical Agents are directly sent back to the Strategic Agent. In this example the system includes Operational Agents. The optimisation function assigned to the Operational Agents is the one described in chapter 5, section 5.3.5.

Reader	*Start Time*	*Finish Time*	*Slack*
Alan	602	607	23
Carla	555	585	75
Flavio	590	595	30
Ian	570	660	27
Nelson	570	580	66
Suresh	628	638	2

Table 7.9: The information sent by the SA to RTA_{sco}

corresponding Operational Agent (OA) responsible for an individual machine. Operational Agents are responsible for optimising the schedule of the operations assigned to them.

The scheduling process followed by the Resource Tactical Agents will be illustrated in detail for the Resource Tactical Agent responsible for the Scotsman. For the other RTAs, only the final results will be presented.

Data sent by the SA to the RTA responsible for the Scotsman (RTA_{sco}) - the SA sends to each RTA the operations to be performed on that resource. The time suggested by the SA for each operation is the earliest time window assigned by the JTA to that operation. Information on the slack allowed for each operation is also sent to the RTAs. Table 7.9 shows the data sent by the SA to the RTA_{sco}.

Generation of graph S_{sco} - figure 7.2 illustrates the graph S_{sco} of the problem that the Resource Tactical Agent responsible for the Scotsman has to solve.

Partitioning of O_{sco} into levels - figure 7.2 illustrates the different levels of graph S_{sco}.

Assignment of operations to individual machines - in order to assign operations to the Operational Agents (in this case two Operational Agents because there are two copies of the Scotsman), the RTA_{sco} partitions the graph S_{sco} into sets of

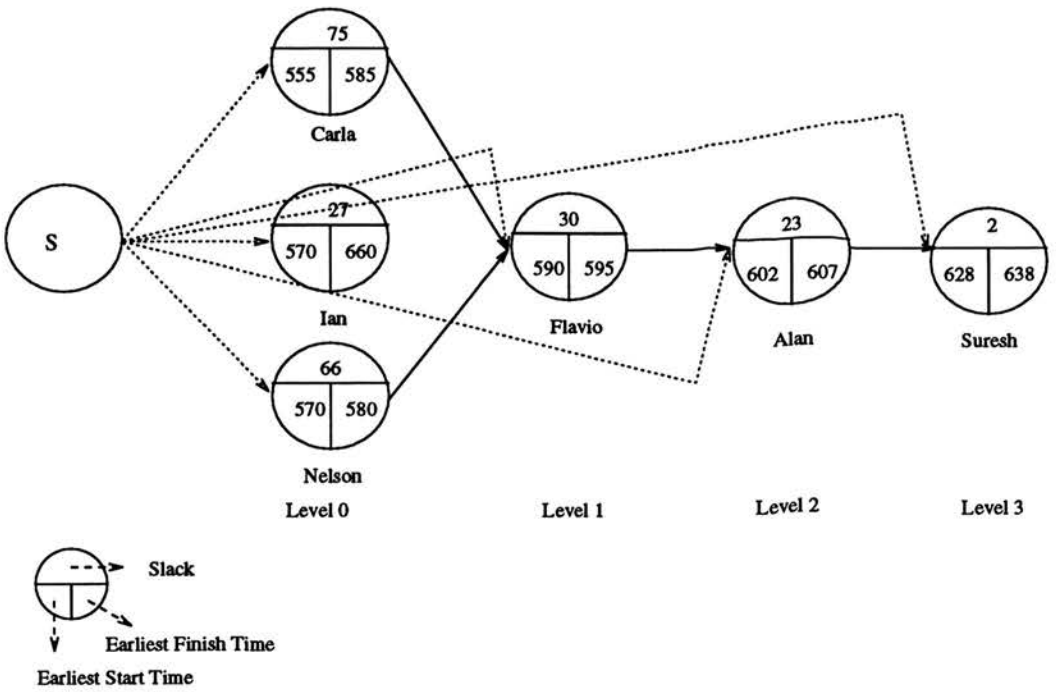

Figure 7.2: The Resource Tactical Agent problem. Graph S_{sco} for the Scotsman

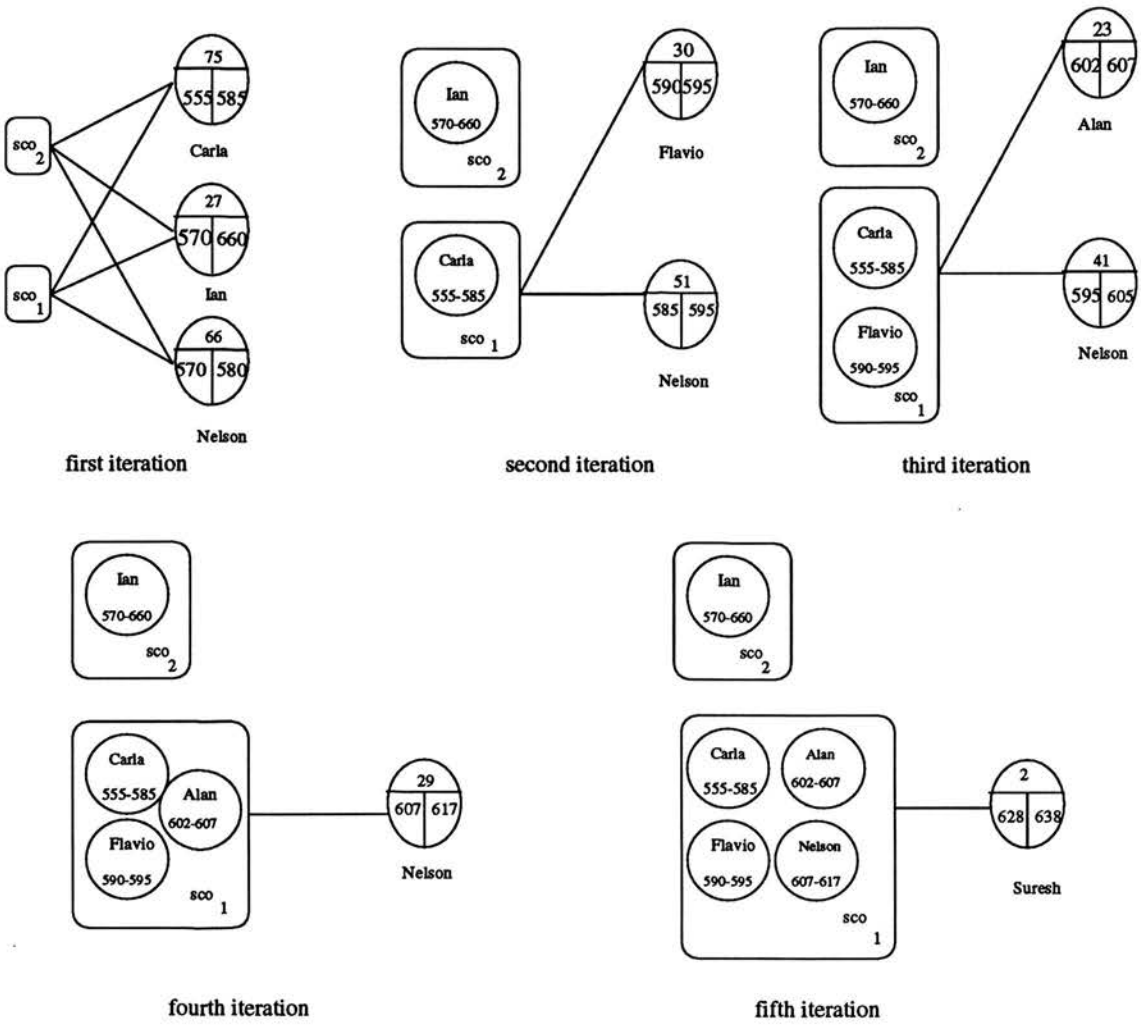

Figure 7.3: The assignment based algorithm for the the Scotsman

operations such that each set of operations corresponds to the set of operations to be performed on the same individual machine with valid times. The RTA_{sco} also assigns times to each operation. This process is performed using the assignment based algorithm (see chapter 5). This algorithm involves at least as many iterations as the number of levels of graph S_{sco}. Let us examine the different steps of the algorithm:

- Construct a graph G^l_{sco} for each level l of the graph S_{sco}. Figure 7.3 illustrates the graphs $G^l_{sco} = (R^l_{sco}, O^l_{sco}, E^l_{sco})$[5], $(l = 0, 1, \ldots, 5)$, for the assignment based algorithm. For example, G^0_{sco} is obtained in the following way:
  - each node of level 0 of graph S_{sco} corresponds to a node of graph G^0_{sco} of the set O^0_{sco}, in this case the operations Carla, Ian and Nelson. Note that this set of nodes might also include nodes that were delayed to the current level. That is the case of the second iteration, for instance, where the operation Nelson (originally from level 0) was delayed to level 1 and therefore it becomes a node of the graph associated with the nodes of level 1.
  - each node of graph G^0_{sco} of the set R^0_{sco} represents an individual machine (in this case *sco*1 and *sco*2) and the set of operations assigned to it. In this case, since it is the first iteration, no operations were assigned to the machines.
  - each arc of set E^0_{sco} links nodes of R^0_{sco} with nodes of the O^0_{sco}. An arc between node R (R $\in R^0_{sco}$) and node (O $\in O^0_{sco}$) means that the operation represented by the node O can be assigned to the machine represented by the node R, with a given start time and finish time, considering the duration of the operation on that machine and considering all the constraints and all the previous operations already assigned to that machine.
- Solve an assignment problem (AP) on graph G^l_{sco}, for each level l. Recall the definition of each assignment problem (AP), considering a generic level l (see below)

[5]Since in this case there is no possibility of confusion between the subscript and superscript, we will use the notation O^l_{sco} for the operations, instead of O^{sco}_l, as used in chapter 5.

- Delay the operations belonging to the level l that were not assigned to any machine to the next level. To delay a node is to shift the earliest start time of the corresponding operation to a later time. The new earliest start time of the operation is the earliest finish time of one of the machines, i.e., the finish time of the operation assigned to a machine in the previous iterations that finishes earliest, and no other operation was assigned to the same machine afterwards. In the case of the first iteration operation Nelson is delayed to level 1 since it was not assigned to any of the machines. The new earliest start time of operation Nelson corresponds to the time the first machine becomes free, i.e., 585.

- Stop condition - all the operations are assigned to a machine.

The AP is defined as :

$$MaxZ = \sum_{i=1}^{q}\sum_{j=1}^{q} u_{ij}x_{ij} \tag{7.1}$$

such that :

$$\sum_{i=1}^{q} x_{ij} = 1 \tag{7.2}$$

$$\sum_{j=1}^{q} x_{ij} = 1 \tag{7.3}$$

$$x_{ij} = 0,1 \qquad (i = 1,2,\ldots r; j = 1,2,\ldots r) \tag{7.4}$$

Where:

- RTA_{sco} - Resource Tactical Agent responsible for the aggregate resource *sco.*
- l - the level of the graph for which the AP is defined
- $k^{(l)}$ - the number of operations of level l that RTA_{sco} has to assign to its machines
- p - the number of individual machines under the responsibility of RTA_{sco}
- q - the maximum of $k^{(l)}$ and p

- op_{il} - the operation i of level l, $(i = 1, 2, \ldots k^{(l)})$
- op_{il} - fictitious operation[6] i of level l, $(i = k^{(l)} + 1, \ldots q)$, if $q > k^{(l)}$
- m_j - the individual machine j, $(j = 1, 2, \ldots p)$
- m_j - fictitious individual machine j, $(j = p + 1, \ldots q)$, if $q > p$
- u_{ij} - the utility associated with the assignment of op_{il} to m_j , where:
  - u_{ij} = large negative value if $i > k^{(l)}$
  - u_{ij} = large value negative if $j > p$
  - u_{ij} = large negative value if op_{il} cannot be assigned to m_j (violation of constraints).
- x_{ij} - decision variable, $(i = 1, 2, \ldots q; j = 1, 2, \ldots q)$

$$x_{ij} = \begin{cases} 1 & \text{if } op_{il} \text{ is assigned to } m_j \text{ with a valid start time} \\ 0 & otherwise \end{cases}$$

Equation 7.1 states that the objective is to maximise the total utility of the assignment. Equation 7.2 guarantees that one (exactly one) operation is assigned to each machine. Equation 7.3 guarantees that each operation is assigned to one (exactly one) machine. Equation 7.4 defines the decision variables.

The definition of the utility function that was adopted in the current implementation of EXPLICIT is as follows:

u_{ir} - utility associated with the assignment of op_{il} to m_r

$$u_{ir} = (1/s_{ilr} + d_{ilr} + g_{ilr}) \tag{7.5}$$

Where:

[6]The introduction of fictitious variables is due to the condition required by the assignment problem formulation that states that the matrix of variables has to be square. Note that the utility values associated with these variables are large negative values.

- $1/s_{ilr}$ - the inverse of the relative slack of operation i of level l on resource r, where[7]:
  -

$$s_{ilr} = \frac{\sum_{j=1,j\neq i}^{k^{(l)}} \{(slackop_{il} + gap_{ilr}) - (duration_{jlr} + gap_{jlr})\}}{\sum_{m=1}^{p} \sum_{v=1}^{k^{(l)}} \sum_{j=1,j\neq v}^{k^{(l)}} \{(slackop_{vl} + gap_{vlm}) - (duration_{jlm} + gap_{jlm})\}}$$

  - $slackop_{il}$ - slack of the operation i of level l, i.e., the interval of time that the operation i of level l can be delayed without delaying the job to which it belongs;
  - $duration_{ilr}$ - duration of the operation i of level l on resource r
  - gap_{ilr} - gap of the operation i of level l on resource r, i.e., the interval of time between the time resource r is available to process an operation and the time operation i of level l is ready to be processed.
  - p - number of individual machines
  - $k^{(l)}$ - number of operations of level l

- d_{ilr} - the relative duration of the operation i of level l on resource r where:
  -

$$d_{ilr} = \frac{duration_{ilr}}{\sum_{m=1}^{p} \sum_{j=1}^{k^{(l)}} duration_{jlm}}$$

  - $duration_{ilr}$ - duration of the operation i of level l on resource r
  - p - number of individual machines
  - $k^{(l)}$ - number of operations of level l

- g_{ilr} - the relative duration of the operation i of level l over the gap on resource r where :

[7]If the minimum value of the component corresponding to the numerator of s_{ilr} is negative ($i = 1, 2, \ldots, k^{(l)}; r = 1, 2, \ldots, p$), a scaling operation is performed transforming it into +1. The same scaling is applied to the corresponding components of the other s_{ilr} ($i = 1, 2, \ldots, k^{(l)}; r = 1, 2, \ldots, p$), and consequently to the denominator of each s_{ilr}, a constant that is obtained by summing all the components that correspond to the numerators of all the s_{ilr}.

–

$$g_{irl} = \frac{\frac{duration_{ilr}}{duration_{ilr}+gap_{ilr}}}{\sum_{m=1}^{p}\sum_{j=1}^{k^{(l)}}\frac{duration_{jlm}}{duration_{jlm}+gap_{jlm}}}$$

- $duration_{ilr}$ - duration of the operation i of level l on resource r
- gap_{ilr} - gap of the operation i of level l on resource r, i.e., the interval of time between the time resource r is available to process an operation and the time operation i of level l is ready to be processed.
- p - number of individual machines
- $k^{(l)}$ - number of operations of level l

Equation 7.5 represents the total utility that is associated with each arc of the graph of RTA_{sco}, for a given level l, G^l_{sco}.

If we consider that with each arc of the graph G^l_{sco} we associate a utility, the final objective is to maximise the total utility that follows from the selected assignment.

To illustrate the way utilities are associated with each arc of the graph G^l_{sco}, let us calculate the utilities for each arc of the graph for the first iteration (level 0). In this example, because the readers take the same time to read a newspaper, independently of the copy that he/she reads and because both copies of the newspaper are available at the same time (525), the components for both machines (copies of the Scotsman) are equal, i.e., $d_{i01} = d_{i02}$, $g_{i01} = g_{i02}$ and $1/s_{i01} = 1/s_{i02}$. This means that, in this case, the crucial issue is to decide which operations are going to be assigned to the machines. Once the operations are selected, it is irrelevant to assign operation, say a, to machine 1, and operation b to machine 2, or operation a to machine 2 and operation b to machine 1. It is a multiple optimal solution.

- Component d_{i0r} - Table 7.10 illustrates the calculation of the component d_{i0r}, d_{i01} and d_{i02}. The underlying idea of this component is to assign a greater utility to longer operations.

i	*Carla*	*Ian*	*Nelson*	Σ
$duration_{i01}$	30	90	10	130
$duration_{i02}$	30	90	10	130
$\sum$	-	-	-	260
d_{i01}	0.11	0.35	0.04	0.5
d_{i02}	0.11	0.35	0.04	0.5

Table 7.10: The component d_{i0r}, d_{i01} and d_{i02},of the utility associated with the arcs of graph G_{sco0} between each operation and *sco*1 and *sco*2

- Component g_{i0r} - Table 7.11 illustrates the calculation of the component g_{i0r}. This component takes into consideration the idle time associated with the assignment of each operation to a given machine. Longer operations that require less idle time are assigned a greater utility.

	Carla	*Ian*	*Nelson*	$\sum$
(1) = $duration_{i01}$	30	90	10	-
(2) = gap_{i01}	30	45	45	-
(3) = $duration_{i01} + gap_{i01}$	60	135	55	-
(4) = (1)/(3)	0.5	0.67	0.18	1.35
(5) = $duration_{i02}$	30	90	10	-
(6) = gap_{i02}	30	45	45	-
(7) = $duration_{i02} + gap_{i02}$	60	135	55	-
(8) = (5)/(7)	0.5	0.67	0.18	1.35
Σ	-	-	-	2.7
g_{i01}	0.18	0.25 0	0.07	0.5
g_{i02}	0.18	0.25 0	0.07	0.5

Table 7.11: The component g_{i0r}, g_{i01} and g_{i02},of the utility associated with the arcs of graph G_{sco0} between each operation and *sco*1 and *sco*2

- Component $1/s_{i0r}$ - Tables 7.12, 7.13 and 7.14 illustrate the calculation of the component $1/s_{i0r}$, $1/s_{i01}$ and $1/s_{i02}$. This component evaluates the remaining slack for a certain operation assuming that it waits for the execution of the other operations on that given resource. Operations whose remaining slack is smaller are assigned a greater utility. This component has the most important weight in the utility function and so it tends to be the decisive component. The other two components tend to be important only as tie-breakers.

	Duration	*Slack*	*Gap*	*Slack + Gap*	*Duration + Gap*
Carla	30	75	30	105	60
Ian	90	27	45	72	135
Nelson	10	66	45	111	55

Table 7.12: Auxiliary table for calculation of the component $1/s_{i0r}$, $1/s_{i01} = 1/s_{i02}$, of the utility associated with the arcs of the graph G_{sco0} between each operation and *sco*1 and *sco*2

	$\{(slackop_{i0} + gap_{i01}) - (duration_{j01} + gap_{j01})\}$			
operation i	Carla	Ian	Nelson	$\sum$
Carla	–	-30	50	20
Ian	12	–	17	29
Nelson	51	-24	–	27
Σ				76

Table 7.13: The numerator of the components s_{i01} and s_{i02}

- Total utility assigned to each arc of graph G_{sco0} - Table 7.15 shows the total utility assigned to each arc of the graph G_{sco0}, $(d_{i0r} + g_{i0r} + 1/s_{i0r})$, $r = 1, 2$. The best assignment chooses the readers Carla and Ian to be assigned to sco1 and sco2. Which reader is assigned to each newspaper is irrelevant in this case (i.e., the value of the solution is equal[8].) Operation Nelson is not assigned to any machine and therefore it is delayed to level 1.

Figure 7.3 on page 131 illustrates the 5 iterations used by the Tactical Agent to delegate operations to each Operational Agent. Tables 7.16 and 7.17 synthesise the assignment performed by the RTA responsible for the Scotsman.

[8]Multiple optimal solution.

machine	$\sum_{j=1,j\neq i}^{k(0)}\{(slackop_{i0}+gap_{i0r})-(duration_{j0r}+gap_{j0r})\}$			$\sum$
	Carla	Ian	Nelson	
$sco1$	20	29	27	76
$sco2$	20	29	27	76
Σ	-	-	-	152
s_{i01}	0.13	0.19	0.18	0.5
s_{i02}	0.13	0.19	0.18	0.5
$1/s_{i01}$	7.69	5.26	5.55	-
$1/s_{i02}$	7.69	5.26	5.55	-

Table 7.14: The component $1/s_{i0r}$, $1/s_{i01}$ and $1/s_{i02}$, of the utility associated with the arcs of graph G_{sco0} between each operation and $sco1$ and $sco2$

	Carla	*Ian*	*Nelson*
sco1	7.98	5.86	5.66
sco2	7.98	5.86	5.66

Table 7.15: Utility assigned to the arcs of the graph G_{sco0} between each operation and $sco1$ and $sco2$

Reader	*Start Time*	*Finish Time*
Carla	555	585
Flavio	590	595
Alan	602	607
Nelson	607	617
Suresh	628	638

Table 7.16: The readers assigned to copy number 1 of the Scotsman by RTA_{sco}

Reader	*Start Time*	*Finish Time*
Ian	570	660

Table 7.17: The readers assigned to copy number 2 of the Scotsman by RTA_{sco}

The Operational Agents

Each Operational Agent is responsible for locally optimising the schedule proposed by the Resource Tactical Agent. The algorithm provided to each Operational Agent minimises maximum lateness of the jobs assigned to it [McMahon & Florian 75][9]. It is an implicit enumeration algorithm that uses a branch-and-bound technique. This algorithm is discussed in detail in appendix A.

In the case of the Scotsman, thereare two Operational Agents (OAs): the OA responsible for copy number 1 (OA_{sco1}) and the OA responsible for copy number 2 (OA_{sco2}). Since only one reader was assigned to OA_{sco2}, there is no optimisation process for OA_{sco2} . Table 7.18 shows the information sent by the RTA_{sco} to OA_{sco1}. Table 7.19 shows the new times assigned by OA_{sco1}. In the new assignment performed by the OA_{sco1}, Nelson reads the Scotsman at 585, instead of 607 as proposed by the RTA, and the reader Flavio reads the Scotsman at 595, instead of 590 as proposed by the RTA. This solution corresponds to the first solution generated by OA_{sco1}, i.e., the first node of the tree of the solutions, since its lower bound (-2) equals the value of the solution for that node.

Reader	*Earliest Start Time*	*Duration*	*Due Date*
Alan	602	5	630
Carla	555	30	660
Flavio	590	5	625
Nelson	570	10	646
Suresh	628	10	640

Table 7.18: The information sent by RTA_{sco} to OA_{sco1}

The other RTAs (and OAs) associated with the other newspapers schedule their readers using a scheduling process identical to the one described for the Scotsman. Table 7.20 shows the final times and levels assigned to the readers by the RTAs (and OAs). This information is sent to the SA.

[9] $L_i = C_i - d_i$ where L_i - lateness of job J_i, C_i - completion time of job J_i, d_i - due date of job J_i.

Reader	*Start Time*	*Finish Time*
Carla	555	585
Nelson	585	595
Flavio	595	600
Alan	602	607
Suresh	628	638

Table 7.19: The times assigned to copy number 1 of the Scotsman by OA_{sco1}

Job	*Tactical Agent*							
	SCO	*level*	*FIN*	*level*	*GUA*	*level*	*EUR*	*level*
Alan	602-607	2	510-570	0	570-600	1	600-602	3
Carla	555-585	0	570-580	0	530-545	0	525-530	0
Ian	570-660	0	660-661	2	661-662	2	662-663	4
Flavio	595-600	1	600-660	1	600-605	1	595-600	2
Nelson	585-595	0	540-570	0	600-610	1	580-584	1
Suresh	628-638	3	603-628	1	525-600	0	600-603	3

Table 7.20: The Resource Tactical Agents' schedules

7.3.2 Iteration 2

Analysis of the Problem (Detection of Conflicts) (2)

The Strategic Agent

The SA detects the conflicts generated from the independent assignment of times performed by each RTA (type g - *conflict generators*). *Conflict generators* occur whenever the start time assigned to an operation by the corresponding Resource Tactical Agent is different from the start time proposed for that operation by the Strategic Agent, i.e., $job - assign - st \neq proptime$ in terms of the attributes of conflicts.

The list of new conflict generators is as follows (note that in each *conflict generator*, $job - assign - st \neq proptime$):

```
conflict-generators

num 5 job carla res fin job-assign-st 570 job-in-conf yes num-conf 5
proptime 545 propslack 75 newtime 570 ftime 580 newslack 50 type g
```

```
status sb parent-gen 5 conf-gen 5 parent (5) no-children cur-it 1

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 4
proptime 584 propslack 66 newtime 600 ftime 610 newslack 50 type g
status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it 1

num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 3
proptime 570 propslack 66 newtime 585 ftime 595 newslack 51 type g
status sb parent-gen 3 conf-gen 3 parent (3) no-children cur-it 1

num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) no-children cur-it 1

num 1 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 1
proptime 590 propslack 30 newtime 595 ftime 600 newslack 25 type g
status sb parent-gen 1 conf-gen 1 parent (1) no-children cur-it 1
```

Where:

```
num           - the reference number of the conflict
job           - the job name
res           - the aggregate resource name
job-assign-st - the start time assigned to the job by the corresponding
                RTA
job-in-conf   - is the job involved in a conflict?
num-conf      - the reference number of the conflicts affecting the job
                involved in this conflict
proptime      - the time proposed for the operation by the Strategic
                Agent
propslack     - the slack proposed for the operation by the Strategic
                Agent
newtime       - the new time proposed for the operation
ftime         - the new finish time proposed for the operation
newslack      - the new slack proposed for the operation
type          - the type of the conflict: g - conflict generator; cr -
                chain reaction conflict; lr - level reaction conflict
status        - the status of the conflict: new; sent; cancelled;
                solved
parent-gen    - the reference number of the root conflict that
                originated this conflict
conf-gen      - the reference number of the immediate conflict
                generator that originated this conflict
parent        - the reference numbers of the immediate parents
                conflicts of this conflict (a conflict might have
                several parents)
```

```
children      - the reference numbers of the conflict children of this
                conflict (no-children is used to indicate 0 children)
cur-it        - the iteration on which the conflict was detected
```

Assignment of Time Windows To Operations (2) - Propagation of Conflicts

The Job Tactical Agents

The Strategic Agent sends the conflict generators to the corresponding Job Tactical Agents. In this case, the JTAs involved in conflicts are: Carla, Flavio and Nelson. The other JTAs (Ian and Suresh) do not have any conflicts on their jobs.

Each Job Tactical Agent propagates the conflicts on its job (type cr - chain reaction conflicts). Appendix B describes the results of conflict propagation per job, and considering two stages:

- before-conflict-resolution, i.e., propagation of conflicts *before* the Strategic Agent has removed conflicts between conflicts
- after-conflict-resolution, i.e., propagation of conflicts *after* the Strategic Agent has removed the conflicts between conflicts (case *conflict generators* vs. *chain reaction conflicts*)

As an example, let us examine the conflict propagation on job Nelson (see appendix B, on page 250, "job-propagation-before-conflict-resolution"). Recall that Nelson's requirements in terms of his reading are (see table 7.1 on page 122):

- 1st - the Financial Times (30 minutes)
- 2nd - the Scotsman (10 minutes)
- 3rd - the European (4 minutes)
- 4th - the Guardian (10 minutes)

There are two *conflict generators* affecting operations of job Nelson:

- Conflict number 4 involves the operation that corresponds to the reading of the Guardian by Nelson. The start time proposed for the operation by the Strategic Agent (*proptime*) was 584. The start time assigned to it by the corresponding Resource Tactical Agent (*newtime*) was 600.

- Conflict number 3 involves the operation that corresponds to the reading of the Scotsman by Nelson. The start time proposed for the operation (*proptime*) by the Strategic Agent was 570. The start time assigned to it by the corresponding Resource Tactical Agent (*newtime*) was 585.

The conflict propagation of conflict number 3 affects the remaining operations of job Nelson after the operation involved in conflict number 3 (the reading of the Scotsman), i.e., the reading of the European and the Guardian by Nelson:

- Conflict number 11 - it is a *chain reaction conflict*, a child of conflict number 3. It involves the operation of the job Nelson after the operation involved in conflict number 3 (the Scotsman) - the reading of the European. The start time proposed for this operation by the Strategic Agent was 580. However, since the reading of the Scotsman was delayed and its new finish time (*ftime*) is 595, the new start time assigned to the reading of the European is 595. Note that the new slack proposed for the operation the European (*newslack*) is 51, while the initial proposed slack (*propslack*) was 66. Conflict number 11 has a child, conflict number 12.

- Conflict number 12 - it is a *chain reaction conflict*, a child of conflict number 11. It involves the operation of the job Nelson after the operation involved in conflict number 11 (the European) - the reading of the Guardian. The start time proposed for this operation by the Strategic Agent was 584. However, since the reading of the European was delayed and its new finish time (*ftime*) is 599, the new start time assigned to the reading of the Guardian is 599. Conflict number 12 does not have any children since it is the last newspaper that Nelson wants to read.

Conflict number 4 involves the reading of the Guardian by Nelson. Since this is the last newspaper to be read by Nelson it does not originate new *chain reaction conflicts*, and so it has "no-children".

Conflicts between conflicts, i.e, conflicts involving the same operation, are detected and solved by the Strategic Agent.

The Strategic Agent's Plan (2)

The Strategic Agent

At this stage the SA possesses all the results of the conflict propagation.

The Strategic Agent generates the *level reaction conflicts* (type lr) and solves conflicts between conflicts. *Level reaction conflicts* are generated based on the information sent by each RTA to the SA, concerning the level of each operation. The SA generates the *level reaction* children of each *chain reaction conflict.* Basically, the *level reaction* children of a given *chain reaction conflict* correspond to all the jobs that have to be performed on the same aggregate resource as the one involved in the *chain reaction conflict*, whose level is equal or greater than the level of the job involved in the *chain reaction conflict.* An operation involved in a *level reaction conflict* does not have its times (proposed earliest start time and slack) changed. It just means that it has to be rescheduled.

For instance, conflict number 13 is a *chain reaction conflict* involving the operation that corresponds to the reading of the Scotsman by Carla (see appendix B, on page 251, "plan-with-lrs"). This operation has level 0 (see figure 7.2 on page 130 and table 7.20 on page 141). The *level reaction conflict* children of conflict number 13 (see the slot children of conflict number 13) correspond to all the operations on the same resource as the operation involved in conflict number 13 (i.e., the Scotsman) and that have level greater or equal than the level of the operation involved in conflict number 13 (i.e., level 0):

- Conflict number 18 - it is a *level reaction conflict* child of conflict number 13. It involves the reading of the Scotsman by Ian. This operation has level 0 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

- Conflict number 17 - it is a *level reaction conflict* child of conflict number 13. It involves the reading of the Scotsman by Nelson. This operation has level 0 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

- Conflict number 16 - it is a *level reaction conflict* child of conflict number 13. It involves the reading of the Scotsman by Flavio. This operation has level 1 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

- Conflict number 15 - it is a *level reaction conflict* child of conflict number 13. It involves the reading of the Scotsman by Alan. This operation has level 2 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

- Conflict number 14 - it is a *level reaction conflict* child of conflict number 13. It involves the reading of the Scotsman by Suresh. This operation has level 3 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

Another example of a *chain reaction conflict* and its *level reaction conflict* children is conflict number 6 that involves the operation that corresponds to the reading of the Scotsman by Flavio (see appendix B, on page 251, "plan-with-lrs"). This operation has level 1 (see figure 7.2 on page 130 and table 7.20 on page 141). Its *level reaction conflict* children correspond to the operations on the same resource that have level equal or greater than 1 (see slot children of conflict number 6)[10], i.e.:

[10] Conflict number 7 is a *chain reaction conflict* child of conflict number 6.

- Conflict number 24 - it is a *level reaction conflict* child of conflict number 6. It involves the reading of the Scotsman by Alan. This operation has level 2 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

- Conflict number 23 - it is a *level reaction conflict* child of conflict number 6. It involves the reading of the Scotsman by Suresh. This operation has level 3 (see figure 7.2 on page 130 and table 7.20 on page 141). Note that the *newtime* of the operation is equal to the *proptime*.

After generating the level reaction conflicts, the Strategic Agent has to solve conflicts between conflicts. The same operation cannot be involved in more than one conflict in the final plan.

To elucidate the way in which the SA elaborates its final plan for conflict resolution, the several intermediate stages of the plan are shown in appendix B. The "plan-with-lrs" is the plan after the generation of the level reaction conflicts. At this stage, all the conflicts between conflicts still exist. The "plan-with-lrs-reds" is the plan with all the conflicts between conflicts removed, except redundant level reaction conflicts. The "final-plan" is the final plan for conflict resolution, after all the conflicts between conflicts have been removed. In this final plan, no two different conflicts can be active on the same operation. The final plan is ordered by increasing proptime.

The conflicts between conflicts, involving the same operation, can be categorised into 4 types:

- *Chain reaction conflicts* vs. *conflict generators* - this situation occurs in the job Nelson and the job Flavio. For instance, in the job Nelson (see appendix B, on page 250, "job-propagation-before-conflict-resolution"), conflict number 4, a *conflict generator*, and conflict number 12 , a *chain reaction conflict*, involve the same operation, the reading of the Guardian by Nelson. According to conflict number 4, Nelson should start reading the Guardian at 600 (*newtime*). According to conflict number 12, Nelson should start reading the Guardian at 599 (*newtime*). Conflict number 4 dominates conflict number 12, since conflict num-

ber 4 imposes a later time for the start time of the reading of the Guardian than the one associated with conflict number 12. Conflict number 12 is cancelled due to the domination of conflict generator number 4 over it (see appendix B, on page 250, "job-propagation-after-conflict-resolution"). In job Flavio, *conflict generator* number 1 is cancelled due to the domination of *chain reaction conflict* number 6 over it.

- *Level reaction conflicts* vs. *conflict generators* - in this situation, the conflict generator always dominates the *level reaction conflict* in terms of the new times that are proposed for the operation that is involved in the conflicts. The *level reaction conflict* is cancelled and the type of the *conflict generator* is changed to type *chain reaction*[11]. In appendix B, on page 251, plan-with-lrs still contains the conflicts between *level reaction conflicts* and *conflict generators* while the plan-with-lrs-reds is expurgated of the corresponding dominated conflicts. For instance, this situation occurs with conflict number 3 and conflict number 17, both involving the operation that corresponds to the reading of the Scotsman by Nelson. The *newtime* proposed for the operation that corresponds to the reading of the Scotsman by Nelson is 585 according to *conflict generator* number 3. The *newtime* proposed for the same operation by *level reaction conflict* number 17 corresponds to the old time of that operation, 570 (recall that operations involved in *level reaction conflicts* do not have their times changed). *Chain reaction conflict* number 3 dominates *level reaction conflict* number 17. Conflict number 17 is cancelled. Conflict number 3, a *conflict generator*, has its type changed to *chain reaction conflict* (see plan-with-lrs-reds).

- *Level reaction conflicts* vs. *chain reaction conflicts* - *chain reaction conflicts* always dominate *level reaction conflicts*, since they propose a more updated time for an operation. When a *level reaction conflict* and a *chain reaction conflict* are active on the same operation, the *level reaction conflict* is cancelled. This rule guarantees the propagation of conflicts. In appendix B, plan-with-lrs still

[11]This is to guarantee that the conflict is re-sent to the respective RTA for reclassification of its level. By definition, *conflict generators* are not sent again to the respective RTA to be rescheduled. However, if there is a *level reaction conflict* on the same job, it has to be rescheduled.

contains the conflicts between *level reaction conflicts* and *chain reaction conflicts* while the plan-with-lrs-reds is expurgated of the corresponding dominated conflicts. For instance, conflict number 16 and conflict number 22 are cancelled due to this rule. *Chain reaction conflict* number 6 involving the operation that corresponds to the reading of the Scotsman by Flavio proposes 605 as the *newtime* for this operation. *Level reaction conflict* number 16 involves the same operation proposing for it the existing time (i.e., *newtime* = *proptime*), 590. *Chain reaction conflict* number 6 dominates *level reaction conflict* number 16. Conflict number 16 is cancelled. Similarly, *chain reaction conflict* number 7 dominates *level reaction conflict* number 22.

- *Level reaction conflicts* vs. *level reaction conflicts* - different *level reaction conflicts* on the same operation have always the same times and characteristics (except the number), so they are redundant. When this happens only one (randomly choosen) out of the several equivalent *level reaction conflicts* is kept active. The remaining equivalent *level reaction conflicts* are cancelled. In appendix B the final-plan is expurgated of all the redundant conflicts. For instance, in plan-with-lrs-reds, conflicts number 19 vs. 25, 14 vs. 23, 15 vs. 24, 20 vs. 26 and 27 vs. 21 are redundant. Conflicts number 25, 23, 24, 26 and 21 are cancelled. Note that, in the final plan, for instance, conflict number 7, the parent of conflict 25 (in the plan-with-lrs-reds), has got conflict number 19 as a new child to compensate it from the loss of conflict number 25. Conflict 19 has two parents: conflict number 7 and conflict number 11.

After removing all the conflicts between conflicts, the Strategic Agent's plan does not contain any two conflicts involving the same operation (see the "final-plan" in the appendix B, on page 256).

Conflict Resolution (2)

The Strategic Agent

After elaborating a plan for conflict resolution the SA possesses all the information for rescheduling. (see the "final-plan" in the appendix B, on page 256).

The RTA that is involved in the *chain-reaction conflict* or *level reaction conflict* with the earliest proptime is the first aggregate resource to be rescheduled. In this case it is the RTA responsible for the Scotsman (see conflict number 13 in the final-plan in the appendix B, on page 256). All the conflicts in the Strategic Agent's plan that involve operations to be performed on the same resource are sent to corresponding Resource Tactical Agent, in this case RTA_{sco}. The Table 7.21 shows the new operations' time windows sent by the SA to the RTA responsible for the Scotsman. This information corresponds to the following conflicts from the final plan:

```
num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 17
num-conf 3 proptime 570 propslack 66 newtime 585 ftime 595 newslack 51
type cr status sent parent-gen 3 conf-gen 3 parent (3) children (11)
cur-it 1

num 6 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 16
num-conf 6 num-conf 1 proptime 590 propslack 30 newtime 605 ftime 610
newslack 15 type cr status sent parent-gen 2 conf-gen 2 parent (2)
children (14 15 7) cur-it 1

num 13 job carla res sco job-assign-st 555 job-in-conf yes num-conf 13
proptime 555 propslack 75 newtime 580 ftime 610 newslack 50 type cr
status sent parent-gen 5 conf-gen 5 parent (5) children (18 15 14)
cur-it 1

num 14 job suresh res sco job-assign-st 628 job-in-conf yes num-conf
23 num-conf 14 proptime 628 propslack 2 newtime 628 ftime 638 newslack
2 type lr status sent parent-gen 5 conf-gen 5 parent (6 13)
no-children cur-it 1

num 15 job alan res sco job-assign-st 602 job-in-conf yes num-conf 24
num-conf 15 proptime 602 propslack 23 newtime 602 ftime 607 newslack
23 type lr status sent parent-gen 5 conf-gen 5 parent (6 13)
no-children cur-it 1
```

```
num 18 job ian res sco job-assign-st 570 job-in-conf yes num-conf 18
proptime 570 propslack 27 newtime 570 ftime 660 newslack 27 type lr
status sent parent-gen 5 conf-gen 5 parent (13) no-children cur-it 1
```

Reader	*Start Time*	*Finish Time*	*Slack*
Alan	602	607	23
Carla	580	610	50
Flavio	605	610	15
Ian	570	660	27
Nelson	585	595	51
Suresh	628	638	2

Table 7.21: The information sent by the SA to RTA_{sco} (second iteration)

Assignment of Start Times To Operations (2)

The Resource Tactical Agent SCO and its Operational Agents

It would be too repetitive to describe again the entire scheduling process followed by the RTA and OAs associated with the Scotsman. Instead, the final results of their scheduling process is summarised in table 7.22.

Job	*Tactical Agent*	
	SCO	*level*
Alan	615-620	1
Carla	580-610	0
Ian	595-685	0
Flavio	610-615	1
Nelson	585-595	0
Suresh	628-638	2

Table 7.22: RTA_{sco}'s schedule (second iteration)

7.3.3 Iteration 3

Analysis of the Problem (Detection of Conflicts) (3)

The Strategic Agent

The SA analyses the results sent by RTA_{sco} in order to identify which conflicts have been solved and/or if new conflicts were generated. In the new results sent by the RTA, if an operation is not associated with a new *conflict generator*, the conflict in which the operation was involved is solved.

The list of new *conflict generators* is as follows:

```
conflict generators

num 32 job ian res sco job-assign-st 595 job-in-conf yes num-conf 32
num-conf 18 proptime 570 propslack 27 newtime 595 ftime 685 newslack 2
type g status sb parent-gen 5 conf-gen 32 parent (32) no-children
cur-it 2

num 31 job alan res sco job-assign-st 615 job-in-conf yes num-conf 31
num-conf 24 num-conf 15 proptime 602 propslack 23 newtime 615 ftime
620 newslack 10 type g status sb parent-gen 5 conf-gen 31 parent (31)
no-children cur-it 2

num 30 job flavio res sco job-assign-st 610 job-in-conf yes num-conf
30 num-conf 16 num-conf 6 num-conf 1 proptime 605 propslack 15 newtime
610 ftime 615 newslack 10 type g status sb parent-gen 2 conf-gen 30
parent (30) no-children cur-it 2
```

Assignment of Time Windows To Operations (3) - Propagation of Conflicts

The Job Tactical Agents

The Strategic Agent sends the *conflict generators* to the corresponding Job Tactical Agents. In this case, the JTAs involved in conflicts are: Alan, Ian and Flavio. The other JTAs (Carla and Nelson) do not have any conflicts on their jobs.

The result of the propagation of conflicts, after resolution of conflicts between conflicts is shown in appendix B (see section Iteration 3, subsection Conflict Propagation on page 258).

The Strategic Agent's Plan (3)

The Strategic Agent

At this stage the SA revises its existing plan taking into consideration the new conflicts and the conflicts that were solved by the rescheduling of the RTA_{sco}.

The Strategic Agent generates the *level reaction conflicts* (type lr) and solves conflicts between conflicts. The process is identical to the process described for iteration 1. The only difference is the removal of the solved conflicts. The final plan is listed in appendix B (see section Iteration 3, subsection Plans on page 259).

Conflict Resolution (3)

The Strategic Agent

The next RTA that has to be rescheduled is RTA_{eur}. Table 7.23 shows the information sent by the SA to the RTA responsible for the European. This information corresponds to the conflicts:

- number 11 (Nelson)
- number 33 (Flavio)
- number 37 (Ian)
- number 39 (Suresh)
- number 40 (Alan)

Note that the job Carla is not sent to RTA_{eur}, since it is not involved in any conflict in the final plan of the SA.

Reader	*Start Time*	*Finish Time*	*Slack*
Alan	600	602	23
Flavio	615	620	10
Ian	687	688	2
Nelson	595	599	51
Suresh	600	603	2

Table 7.23: The information sent by the SA to RTA_{eur}

Assignment of Start Times To Operations (3)

The Resource Tactical Agent EUR and its Operational Agents

The final results of the scheduling process of the RTA responsible for the European and its OAs are summarised in table 7.24.

Job	*Tactical Agent*	
	EUR	*level*
Alan	600-602	1
Ian	687-688	3
Flavio	615-620	2
Nelson	595-599	0
Suresh	600-603	1

Table 7.24: RTA_{eur}'s schedule (second iteration)

7.3.4 Iteration 4

Analysis of the Problem (Detection of Conflicts) (4)

The Strategic Agent

The SA analyses the results sent by RTA_{eur} in order to identify which conflicts have been solved and/or if new conflicts were generated. In the new results sent by the RTA, if an operation is not associated with a new *conflict generator*, the conflict in which the operation was involved is solved. No new *conflict generators* were detected. It means that all of the conflicts sent to RTA_{eur} were solved.

The Strategic Agent's Plan (4)

The Strategic Agent

At this stage the SA revises its existing plan. The new final plan is listed in appendix B (see section Iteration 4, subsection Plans on page 261).

Conflict Resolution (4)

The Strategic Agent

The next RTA that has to be rescheduled is RTA_{fin}. Table 7.25 shows the information sent by the SA to the RTA responsible for the Financial Times. This information corresponds to the conflicts:

- number 34 (Flavio)
- number 35 (Ian)
- number 42 (Suresh)

Note that the jobs Alan, Carla and Nelson are not sent, since they are not involved in any conflict of the final plan.

Reader	*Start Time*	*Finish Time*	*Slack*
Flavio	620	680	10
Ian	685	686	2
Suresh	603	628	2

Table 7.25: The information sent by the SA to RTA_{fin}

Assignment of Start Times To Operations (4)

The Resource Tactical Agent FIN and its Operational Agents

The final results of the scheduling process of the RTA responsible for the Financial Times and its OAs is summarised in table 7.26.

Job	*Tactical Agent*	
	FIN	*level*
Ian	685-686	1
Flavio	620-680	0
Suresh	603-628	0

Table 7.26: RTA_{fin}'s schedule (second iteration)

7.3.5 Iteration 5

Analysis of the Problem (Detection of Conflicts) (5)

The Strategic Agent

The SA analyses the results sent by RTA_{fin} in order to identify which conflicts have been solved and/or if new conflicts were generated. No new *conflict generators* were detected. That means that all of the conflicts sent to RTA_{fin} were solved.

The Strategic Agent's Plan (5)

The Strategic Agent

At this stage the SA revises its existing plan. The new final plan is listed in appendix B (see section Iteration 5, subsection Plans on page 262).

Conflict Resolution (5)

The Strategic Agent

The next RTA that has to be rescheduled is RTA_{gua}. Table 7.27 shows the information sent by the SA to the RTA responsible for the Guardian. This information corresponds to the conflicts:

- number 36 (Ian)

Note that the jobs Alan, Carla, Flavio and Nelson are not sent, since they are not involved in any conflict of the final plan.

Reader	*Start Time*	*Finish Time*	*Slack*
Ian	686	687	2

Table 7.27: The information sent by the SA to RTA_{gua}

Assignment of Start Times To Operations (5)

The Resource Tactical Agent GUA and its Operational Agents

The final results of the scheduling process of the RTA responsible for the Guardian and its OAs is summarised in table 7.28.

Job	*Tactical Agent*	
	GUA	*level*
Ian	686-687	1

Table 7.28: RTA_{gua}'s schedule (second iteration)

7.3.6 Iteration 6

Analysis of the Problem (Detection of Conflicts) (6)

The Strategic Agent

The SA analyses the results sent by RTA_{gua} in order to identify which conflicts have been solved and/or if new conflicts were generated. No new *conflict generators* were detected. That means that all of the conflicts sent to RTA_{gua} were solved.

The Strategic Agent's Plan (6)

The Strategic Agent

At this stage the SA revises its existing plan. The new final plan is an empty plan. That means the final solution is reached.

Table 7.29 summarises the start and finish times assigned to each reader for each newspaper.

Reader	*Up At*	*Due At*	*1st Newspaper*			*2nd Newspaper*			*3rd Newspaper*			*4th Newspaper*		
Alan	510	630	F	510	570	G	570	600	E	600	602	S	615	620
Carla	525	660	E	525	530	G	530	545	F	570	580	S	580	610
Flavio	585	690	G	600	605	S	610	615	E	615	620	F	620	680
Ian	570	690	S	595	685	F	685	686	G	686	687	E	687	688
Nelson	540	660	F	540	570	S	585	595	E	595	599	G	600	610
Suresh	525	640	G	525	600	E	600	603	F	603	628	S	628	638

Table 7.29: The final solution for the newspaper reading problem

7.4 Summary

In this chapter a simple example was comprehensively described. The main purpose of it was to illustrate the application of the methodology proposed in this thesis, in

particular in terms of the roles performed by the different agents of the system and their interaction. The scheduling process adopted in EXPLICIT was described focusing on the interaction among agents and on the role of the Strategic Agent in terms of conflict detection and conflict resolution.

Chapter 8

Comparison with other Systems

A comparison of the approach adopted in EXPLICIT with other approaches is presented in this chapter. Particular emphasis is given to the comparison between EXPLICIT and DAS, since EXPLICIT is mainly inspired by DAS's architecture.

8.1 DAS

8.1.1 Overview

DAS (Distributed Asynchronous System) was developed at the University of Strathclyde [Buchanan *et al* 88], [Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89]. DAS is a logically distributed system though implemented in a sequential machine. The structure of DAS is a hierarchy of units. Each unit within the hierarchy has an attached agent. The scheduling problem is decomposed across the hierarchy of communicating agents. The external world is treated as an agent with one exceptional property, negotiation is not allowed. Because of the asynchronous nature of the decision making process, the system relies heavily on the communication facilities. Additionally, the constraint propagation mechanism provides a basis for the passage of messages across the hierarchy. There is a priority mechanism that allows individual agents to maintain beliefs that are temporarily inconsistent with the global hypothesis. DAS does not differentiate between prediction and reaction.

8.1.2 DAS Components

There are 4 component parts to DAS.

The structural representation

- Resources
  - Top level - a single unit represents a holistic view of the scheduling problem (Strategic Level)
  - Middle level - a unit represents an aggregation of similar resources (Tactical Level)
  - Lowest level - a unit represents an individual resource (Operational Level)
- Operations - a normal operation represents an individual process which must be processed as part of a job. Special types of operations are used to represent special events such as the unavailability of resources due to failure or maintenance tasks.
- Plans - the work to be put through the production facilities is represented by a process plan. A process plan is a set of operations, connected by temporal precedence relations. The process plan describes the route to be followed by the different operations within the job . Operations are assigned to a type of resource (but not to an individual resource). Non-linear plans are allowed.

The active representation of the schedule

The current schedule (the global hypothesis), is a binding of operations to resources over time. The current global hypothesis is implicitly defined by: the operations (start times); the resources (allocated to operations) and the constraints attached to them (originated from scheduling decisions and environment events). An active representation of the global hypothesis is used. A constraint maintenance system is responsible for the propagation of any change made to the global hypothesis, such as the binding of an operation to a point in time, through a process plan, and the notification of the change is sent to the affected agents. The role of the constraint

maintenance system is to guarantee that both the scheduling decisions of agents and relevant events from the environment are recorded in the global hypothesis, highlighting conflicts between problem solving agents by making the consequences of all known constraints explicit to all the agents.

The problem solving agents

- Strategic Agent - this agent is attached to the strategic unit. It is responsible for releasing of work into the system and for conflict resolution. The Strategic Agent's mechanisms to solve conflicts are backtracking and temporal constraint relaxation.

- Tactical Agent - this agent is attached to a unit representing a tactical resource. It performs a load balancing role. It is responsible for delegating and retracting work to and from its subordinates (technological constraint satisfaction). Each Tactical Agent has its T-assistant. The T-assistant performs a clerical role for its Tactical Agent. It records all the decisions its Tactical Agent has made and the consequences of these decisions. Each Tactical Agent always consults with its T-assistant for advice before making any decision. If a Tactical Agent has an over constrained problem it notifies the superior Strategic Agent by sending it an inter-resource conflict set.

- Operational Agent - this agent is attached to a unit representing an operational resource. It is responsible for maintaining a schedule on an operational resource, i.e., the allocation of start times to its operations (temporal constraints satisfaction). Whenever an Operational Agent is faced with an over constrained problem it computes an explanation of why it believes the problem is over constrained (an intra-resource conflict set) and notifies its superior Tactical Agent.

The mechanism for co-ordinating problem solving effort

The mechanism for focusing problem solving effort is the prioritising of messages. Priorities originate from the act of decision making (the allocation of start times to operations on resources). Priority can be considered as a history of problem solving

difficulty or a measure of criticality. This mechanism allows the Operational Agents to maintain beliefs that are different from those globally held. Each operation within the system has a priority attribute, initialised to zero. The priority of an operation is a measure of the difficulty associated with the satisfaction of its temporal and technological constraints. Whenever an Operational Agent reports to its superior Tactical Agent that a set of operations is involved in a conflict set, the Tactical Agent increases the priority of the operations involved in the conflict set. To solve the conflict set, the Tactical Agent might retract one or more operations belonging to the conflict set from the respective Operational Agent and assign them to other Operational Agents. When an Operational Agent assigns a start time to an operation, the start time is written out to the global hypothesis and the constraint maintenance system updates the temporal domains of the other operations in the same process plan (i.e., the same job) and a series of messages are sent to the agents that hold the others operations. Basically each message tells the Operational Agents to modify the priority of a given operation (the operation that is in the same process plan as the one that had a new start time assigned to it). The priority of a message corresponds to the priority of the operation that originated the message. When an Operational Agent receives a message to modify the priority of one of its operations, it checks the priority of the message. If the message's priority is greater or equal to the current priority of the operation, the Operational Agent has to modify the priority of the operation. If the message's priority is less than the priority of the operation and its domain has been reduced, the Operational Agent can ignore the message and continue to believe the more relaxed temporal domain of the operation. This mechanism allows an Operational Agent to make a dominant decision within a process plan. Initially all operations in a plan have a priority of zero. It is possible to pre-process plans, attaching a non-zero priority to operations (for example when it is expected that an operation has a demand for a critical resource the priority of that operation can be made high relative to other operations in the plan). Sometimes it can happen that an under-constrained Operational Agent makes a dominant decision before an over-constrained Operational Agent. To break this situation the system resorts to inter-agent backtracking coordinated by the Tactical Agents. When the mechanism of priorities is inadequate a more disciplined

technique is employed either:

- a decision ordering through process plans
- a synchronisation of agents

8.1.3 The Scheduling Process

When a job is introduced to the system the Strategic Agent is notified. The Strategic Agent delegates each operation in the process plan of the job to individual Tactical Agents. Each Tactical Agent then delegates each of its operations to an Operational Agent and informs its T-assistant of this decision. The Operational Agent then attempts to introduce this new operation into its local schedule. If it succeeds, it writes out a start time for that operation initiating constraint propagation through the job's process plan with resultant messages sent to all the agents that hold the remaining operations in the plan. At this point there is no decision ordering through the process plan, so decisions can be made asynchronously. If an Operational Agent cannot produce a consistent schedule it reports to its superior Tactical Agent an intra-resource conflict set, i.e., the set of operations that the Operational Agent believes cannot be consistently scheduled on its resource. The Tactical Agent informs its T-assistant of this conflict set and asks its T-assistant for advice, i.e., what load balancing options remain. The Tactical Agent then load balances, i.e., the Tactical Agent retracts operations from Operational Agents and delegates operations to Operational Agents. If there are no load balancing options available, the Tactical Agent concludes that the problem is over constrained and it delivers an inter-resource conflict set, in disjunctive normal form, to its superior, the Strategic Agent. The Strategic Agent analyses the inter-resource conflict sets received from the Tactical agents and decide upon a course of action, a combination of inter-agent backtracking and constraint relaxation.

8.2 EXPLICIT vs. DAS

EXPLICIT was inspired by DAS. However, the similarities between EXPLICIT and DAS are only at the conceptual level. The main similarities between the two systems are

related to the type of organisational structure adopted for EXPLICIT. The techniques and methods used in both systems are substantially different. EXPLICIT was designed and implemented from scratch. The only source of information in terms of DAS consisted of the papers published by the authors, [Buchanan *et al* 88], [Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89], among others. In this section we are particularly interested in analysing the two systems considering the roles and functions performed by the different agents involved the scheduling process. However, we briefly refer to the techniques used in DAS.

8.2.1 General Considerations

The approach adopted in EXPLICIT aims to integrate multiple scheduling perspectives, to integrate predictive capabilities with reactive capabilities and to increase optimisation rather than settling for satisfaction. The underlying principles to achieve these goals are: the assignment of different functional roles and skills to agents within a distributed problem solver framework; ensuring *pluralism*; the merge of OR and AI techniques; ensuring *contextual awareness*; providing agents with mechanisms to detect and solve *Conflict*.

EXPLICIT and DAS use uniform hierarchies (see chapter 3) with three levels of management: the Strategic Level; the Tactical Level and the Operational Level. In both systems the scheduling process is distributed along the hierarchy of agents. In both systems *Conflict* is allowed. Both systems are (logically) distributed. To some extent, EXPLICIT adopts some of the main concepts embodied in DAS and pushs them one step further. In this way, EXPLICIT complements some of the less strong features of DAS.

The *Organisational* Structure

EXPLICIT and DAS use *uniform hierachies*, with three levels of management. In EXPLICIT the *Organisation* metaphor was pushed one step further. Providing the agents of the system with different functional capabilities was a central guide line to the approach adopted in EXPLICIT. Agents, depending on their position in the hierarchy,

were given different roles. In particular, the functional capabilities of planning and evaluation and analysis were reinforced in the system. Because *conflict* is allowed and even stimulated, planning, analysis and evaluation capabilities are important functions to ensure the *Global Coherence* of the system as a whole. Planning, analysis and evaluation capabilities were assigned to the different agents of the system, in particular to the agents that have delegation functions - the Strategic Agent and the Tactical Agents (see chapter 6).

Pluralism

In DAS the scheduling process is mainly *resource based.* Tactical Agents are organised in terms of resources. Scheduling decisions are performed in terms of resources and then propagated throughout the jobs. In EXPLICIT Job Tactical Agents are introduced together with the Resource Tactical Agents. Job Tactical Agents enforce the *job based perspective* while Resource Tactical Agents enforce a *resource based perspective.* The Strategic Agent is provided with mechanisms to detect the criticality of operations, i.e., to enforce an *operation-based perspective.* EXPLICIT is well provided with mechanisms that ensure *opportunistic*[1] and *micro-opportunistic*[2] scheduling (see chapter 2).

In EXPLICIT *pluralism* is ensured by allowing agents to perform their scheduling decisions independently of each other, without communication with each other.

In EXPLICIT *pluralism* is ensured by providing agents with both empirical knowledge (heuristics, dispatch rules) and theoretical knowledge (optimal algorithms)

Merge of OR and AI techniques

DAS aims at "satisfaction". EXPLICIT aims at increasing "optimisation". EXPLICIT combines OR with AI techniques. EXPLICIT combines optimal models to tackle the well structured (sub)problems assigned to the agents with heuristic style heuristic reasoning

[1]The term is borrowed from [Smith *et al* 86] [Smith 87] since a *job based perspective* is combined with a *resource based perspective.*

[2]The term is borrowed from [Sadeh 91] since the criticality of each operation is analysed individually.

models for the more complex, judgemental and behavioural parts.

Conflict

Conflict is a central concept in EXPLICIT as it is in DAS. *Conflict* arises as a consequence of changes that occur in the environment and also due to scheduling decisions performed by the agents. However, in EXPLICIT, the latter form of *conflict* is stimulated as it is recognised that *conflict* as a consequence of the scheduling process should be exploited as a form of guaranteeing *pluralism*, as a form of guaranteeing that Tactical Agents perform their own decisions without influencing each other. This is a key difference between EXPLICIT and DAS.

In DAS, whenever a start time is assigned to an operation, all the agents involved in the same process plan (job) are notified. Though *conflict* is allowed, the constraint maintenance mechanism attenuates it. The priority mechanism allows agents to ignore some of the notifications, if their operations that are involved in the conflict have a priority greater than the priority of the message, otherwise they are forced to change the priority and times of their operations. In DAS all the agents involved in the same job are aware of the scheduling decisions performed by each other.

In EXPLICIT a quite different approach is adopted. The different Tactical Agents perform their scheduling decisions without being aware of the decisions performed by other Tactical Agents. In particular, Resource Tactical Agents assign times and individual machines to operations (with or without the intervention of the operational Agents), without any form of communication among each other. The only information that the Resource Tactical Agents know about each operation is its time windows. Whenever Resource Tactical Agents, with or without the Operational Agents, assign times and machines to operations, there is no mechanism to immediately propagate that effect throughout the other operations of the same job.

The idea embodied in EXPLICIT is to allow Tactical Agents (Job Tactical Agents and Resource Tactical Agents) to formulate their best scheduling decisions considering their own interests. On the other hand, the Strategic Agent is provided with mechanisms

to analyse and evaluate the criticality of conflicts, the criticality of each operation per se and, based on that, the Strategic Agent defines plans for conflict resolution.

Structural Awareness

In a distributed and conflicting environment it is essential to provide the different agents of the system with a set of mechanisms to make them aware of the structural and intrinsic properties of the (sub)problems that they have to solve and the interaction of their (sub)problems, without relying on communication with each other. The underlying thesis to provide such mechanisms to a system is that the inherent structural properties of a domain can be used to alleviate its "intractability", in the sense that the whole search space is decomposed into smaller search spaces.

In DAS communication among agents is the way to make the different agents of the system aware of each other's scheduling decisions. EXPLICIT provides agents with several mechanisms to make them *structurally aware* of the (sub)problems that they have to solve (see chapter 6).

8.2.2 Problem Solving Agents

The Strategic Agent

Both in EXPLICIT and in DAS the Strategic Agent is responsible for the introduction of new work into the system.

In DAS the interference of the Strategic Agent for conflict resolution is a last resort, since the agents communicate among each other their scheduling decisions. In case of *conflict* the Strategic Agent can perform inter-agent backtracking (dependency directed backtracking [Prosser 88]) and due-date relaxation.

In EXPLICIT, since Tactical Agents are not aware of each other's scheduling decisions, the Strategic Agent is assigned with mechanisms to detect conflict and with an elaborated algorithm for conflict resolution, for inter-agent backtracking and due-date relaxation. The conflict resolution procedure takes into consideration the complexity

experienced by the Tactical Agents, reflected in their measures of *structural awareness.* The Strategic Agent is equipped with a rich taxonomy of conflicts, including definition of dependencies among conflicts and definition of criticality and dominance of conflicts.

The Tactical Agents

A major difference between EXPLICIT and DAS lies in the different roles assigned to the Tactical Agents.

EXPLICIT explicitly includes Job Tactical Agents. Job Tactical Agents enforce the *job-based* scheduling perspective. They are responsible for the elaboration of the initial process plan for their jobs and for replanning (conflict propagation).

DAS is provided with a sophisticated constraint maintenance system in order to ensure that the various agents in the system share a common view of the jobs. The constraint maintenance mechanism makes explicit the implicit effects of constraints added to the system. In DAS, temporal constraint propagation is used to maintain consistency within the constraint networks representing process plans. The form of propagation employed to performed this task has been classified as label inferencing [Davis 87].

In what concerns the Resource Tactical Agents (Tactical Agents according to DAS nomenclature), the difference between the two systems is even greater. In both systems, Resource Tactical Agents (Tactical Agents according to DAS nomenclature) are responsible for load balancing of operations amongst its subordinates resources. The Resource Tactical Agents are responsible for the delegation of the operations to their subordinate Operational Agents. However, the way this operation is performed in both systems is substantially different.

In DAS, Tactical Agents do not have a well defined strategy to delegate work. Tactical Agents only assign individual machines to operations without assigning times to them. The assignment of times is entirely performed by the Operational Agents in DAS. The strategy adopted by the Tactical Agents (in DAS) is trial and error, with the help of the T-assistant performing a clerical role. The Tactical Agent first selects the

next operation to delegate and the machine to delegate it to[3].The Tactical Agent's problem in DAS is a dynamic constraint satisfaction problem [Prosser 88]. The T-assistant views the load balancing of operations as a consistent labelling problem. To record inconsistencies in terms of assignment of machines to operations, the T-assistant uses a reason maintenance system (see [de Kleer 86, Doyle 81] [Burke & Prosser 89, Prosser 88]).

Conversely, in EXPLICIT the Resource Tactical Agents are provided with a quite elaborated mechanism to analyse the complexity of the problems assigned to them, in order to perform the best load balancing of operations on their subordinate resources. In EXPLICIT, it is possible for the system to work only with Resource Tactical Agents without Operational Agents. In fact, in EXPLICIT, not only do Resource Tactical Agents assign machines to operations, they also assign times to the operations. The partitioning of the whole (sub)problem into levels gives each Resource Tactical Agent a notion of the opportunities and conflicts that are associated to the different jobs to be scheduled on its resource. The concept of level provides the Resource Tactical Agents with a mechanism to assign priorities to each job considering the opportunities and conflicts detected. A major mechanism assigned to the Resource Tactical Agents to define goals and to define a plan to assign operations to the lower level (their Operational Agents) is the assignment based algorithm which includes the generalised assignment algorithm (see [Christofides 75, Gondran & Minoux 84], [Gomes 87, Gomes & Almeida 88], [Ball *et al* 81, Ball & Roberts 85]).

The Operational Agent

In DAS Operational Agents are vital for the system. They are responsible for assigning times to the operations delegated by the Tactical Agents. Operational Agents view their problems as dynamic constraint satisfaction problems which operate on constraint graphs [Mackworth 77], [Prosser 88]. The techniques used are forward checking [Haralick & Elliot 80], shallow learning [Detcher 86] and dependency directed back-

[3]The strategy for selecting the next operation is a parameter. However the actual version of the system only considers the strategy "first".

tracking [Gaschnig 77, Stallman & Sussman 77].

In EXPLICIT the existence of Operational Agents is not vital. Operational Agents, if provided to the system, are responsible for improving the schedule suggested by their superior Resource Tactical Agents. In EXPLICIT the Operational Agents are provided with an optimal algorithm that minimises maximum lateness of the jobs assigned to them [McMahon & Florian 75].

Communication Among Agents

As a final remark it is worth emphasising that while in EXPLICIT all the communication between agents flows vertically through the hierarchy (see chapter 6), in DAS there is also communication flowing horizontally and even diagonally between all the different agents involved in the same jobs. In DAS, horizontal and diagonal communication is a mechanism to maintain all the agents involved in the same job informed about the scheduling decisions concerning that job. However, because of this mechanism, some agents can be prevented from making good scheduling decisions due to decisions concerning the same job previously made by other agents. In EXPLICIT there is no horizontal or diagonal communication between agents involved in the same job. This strategy allows each agent to formulate its best scheduling decisions without being constrained by the decisions of the other agents. The Strategic Agent, based on the criticality of the conflicts that occur as a consequence of the asynchronous scheduling process performed by the Tactical Agents, defines which Tactical Agents' scheduling decisions are accepted and which operations have to be rescheduled. This strategy seeks solutions with better quality by allowing all the agents to perform their scheduling decisions independently of each other and so making explicit the opportunities and conflicts associated with each agent's scheduling problem.

8.3 EXPLICIT vs. Others Systems

A general comparison between EXPLICIT and others systems is presented in this section considering the three aspects: integration of multiple scheduling perspectives; reaction

vs. prediction; and satisfaction vs. optimisation.

8.3.1 Integration of Multiple Scheduling Perspectives

ISIS [Fox & Smith 84] and SOJA [Le Pape 85] use essentially a *job based perspective*, while RESS-II [Liu 88], OPT [Jacobs 84] and DAS ([Buchanan *et al* 89], [Burke 89], [Burke & Prosser 89]) are mainly *resource based* schedulers. OPIS [Smith 87] and SONIA [Collinot *et al* 88] interleave a *job based perspective* with a *resource based perspective.* This strategy of scheduling has been named *opportunistic scheduling.* EXPLICIT also interleaves a *job based perspective* with a *resource based perspective.* However, EXPLICIT goes one step further in the sense that, though the *job based perspective* and the *resource based perspective* are combined to analyse the complexity of the scheduling problems, decisions are made considering as focus of attention the criticality of each operation per se (*operation based perspective*). To some extent this approach has some similarities with the approaches adopted by [Sadeh 91] in MICRO-BOSS and by [Berry 91]. [Sadeh 91] referred to this type of scheduling as *micro-opportunistic.* The approaches used by [Sadeh 91] and [Berry 91] consist essentially in the identification of periods of high resource contention estimated by aggregating assumed operations demand. Bottlenecks are managed by allocating operations contributing to the predicted bottleneck to times outside of the bottleneck period. EXPLICIT implements the *operation based perspective* (*micro-opportunistic* according to Sadeh's terminology) performing a sequence of optimisations of the load balancing of the resources, each time assigning times to the operations considered critical and to the operations not involved in conflicts. Each operation is analysed individually (*operation based perspective*) and its criticality results from the combination of a *job based perspective* (*chain reaction conflicts*) with a *resource based perspective* (*level reaction conflicts* and *conflict generators*).

8.3.2 Reaction vs. Prediction

OPT, ISIS, RESS-II, SOJA are *predictive* systems. OPIS, DAS, SONIA, DAS integrate *predictive* capabilities with *reactive* capabilities. These systems claim a common viewpoint

of *predictive* and *reactive* scheduling.

One of the main goals of EXPLICIT was to strengthen the *predictive* capabilities of a *reactive* (or *predictive/reactive*) scheduler, i.e., DAS. So, to some extent EXPLICIT is better equipped in terms of *predictiveness* than in terms of *reactiveness*. However, it should be pointed out that though the approach adopted in EXPLICIT specially aimed to increase the *predictive* capabilities of a *predictive/reactive* scheduler, the idea of *reactive* scheduling is compatible with EXPLICIT. On the contrary, the major features that make DAS be a *reactive* scheduler are kept in EXPLICIT.

8.3.3 Satisfaction vs. Optimisation

EXPLICIT aims to increase "optimisation" rather than settling for "satisfaction" of a problem. Existing schedulers [4] mainly incorporate heuristic procedures. EXPLICIT integrates optimal models to solve the mathematically well defined components of the (sub)problems its agents have to solve with more heuristic type procedures for the more qualitative components of the system. EXPLICIT performs a sequence of optimisations of the load balancing of the resources, each time assigning times to the operations considered critical and to the operations not involved in conflicts. Each operation is analysed individually (*operation based perspective*) and its criticality results from the combination of a *job based perspective* (*chain reaction conflicts*) with a *resource based perspective* (*level reaction conflicts* and *conflict generators*). To some extent, considering the local optimisations performed in each cycle by the Resource Tactical Agents, there are some similarities between EXPLICIT and the "Shifting Bottleneck Procedure" adopted by [Adams *et al* 88] (see also chapter 2).

In the next chapter a case performance analysis of EXPLICIT is presented, as well as a comparison between the results produced by EXPLICIT and certain dispatch rules.

[4] We are excluding particular systems tailored for very particular domains and problems. As mentioned in chapter 2, there is a huge and rich subarea of OR research devoted to finding optimal solutions for particular problems.

8.4 Summary

The approach adopted in EXPLICIT is compared with other approaches, in particular with DAS. To some extent, EXPLICIT adopts some main concepts embodied in DAS and pushes them one step further.

In EXPLICIT, the functional capabilities of analysis, planning and evaluation were reinforced. *Pluralism* is ensured by allowing agents to perform their scheduling decisions independently of each other, by providing agents with both empirical knowledge (heuristics, dispatch rules) and theoretical knowledge (optimal algorithms) and by explicitly allowing the coexistence of a *job based perspective*, a *resource based perspective* and an *operation based perspective*, enabling so called *opportunistic* and *micro-opportunistic* scheduling. This approach is implemented in EXPLICIT by performing a sequence of optimisations of the load balancing of the resources, each time assigning times to the operations considered critical and to the operations not involved in conflicts. Each operation is analysed individually (*operation based perspective*) and its criticality results from the combination of a *job based perspective* (*chain reaction conflicts*) with a *resource based perspective* (*level reaction conflicts* and *conflict generators*).

Conflict is a central concept in EXPLICIT as it is in DAS. However, in EXPLICIT *conflict* as a consequence of the scheduling process is stimulated as a form of guaranteeing *pluralism* and to guarantee that Tactical Agents perform their best scheduling decisions considering their own interests. On the other hand, considering the *conflicting* nature of the environment, EXPLICIT provides agents with several mechanisms to make them *structurally aware* of the (sub)problems that they have to solve, in particular the Strategic Agent is provided with mechanisms to analyse and evaluate the criticality of conflicts, the criticality of each operation per se and, based on that, the Strategic Agent defines plans for conflict resolution.

EXPLICIT was conceived to strengthen the *predictive* capabilities of a *reactive* scheduler, without making *predictiveness* incompatible with *reactiveness*.

EXPLICIT integrates optimal models to solve the mathematically well defined components of the (sub)problems its agents have to solve with more heuristic type procedures

for the more qualitative components of the system. It thus combines OR and AI techniques in a useful way.

Chapter 9

Performance Analysis of EXPLICIT

A case performance analysis is presented in this chapter. The performance analysis compares the results produced by two different versions of EXPLICIT, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, and the results produced by EXPLICIT against the results produced by four dispatch rules. EXPLICIT-OAS-IN, a version of EXPLICIT that includes Operational Agents, performs considerably better than EXPLICIT-OAS-OUT, a version of EXPLICIT without Operational Agents. EXPLICIT outperforms the four dispatch rules, especially in terms of tardiness and utilisation of resources, the two main objectives embodied in the utility functions assigned to the Resource Tactical Agents. This chapter also includes the description of the generator of the battery of cases used for the performance analysis.

9.1 Two Instances of EXPLICIT

To analyse the performance of the general framework proposed for EXPLICIT in chapter 5, two different versions of the system were considered. The two versions are identified by EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN[1], respectively. In this section we detail the characteristics of the two versions.

[1]The rationale for these two acronyms is: both versions are particular instances of EXPLICIT; the first version, EXPLICIT-OAS-OUT, apart from the Strategic Agent, only includes the Tactical Agents (Operational Agents are "out").; the second version, EXPLICIT-OAS-IN, apart from the Strategic Agent, includes the Tactical Agents and the Operational Agents (Operational Agents are "in").

9.1.1 EXPLICIT-OAS-OUT

The version EXPLICIT-OAS-OUT corresponds to the job-shop scheduler described in chapter 5, excluding the Operational Agents. This means that the times assigned to the operations by the RTAs are the ones that are sent back to the Strategic Agent. There is no further optimisation performed by the Operational Agents. In EXPLICIT-OAS-OUT, Operational Agents do not exist.

9.1.2 EXPLICIT-OAS-IN

The version EXPLICIT-OAS-IN corresponds to the job-shop scheduler described in chapter 5, including the Operational Agents. Operational Agents are responsible for optimising the times of the operations assigned to them by the corresponding Resource Tactical Agent.

Several optimisation roles can be assigned to the Operational Agents[2], which, as in the case of the definition of the utility function associated with the APs that the RTAs have to solve, can be used to tune up the system for a particular domain and problem.

In the version EXPLICIT-OAS-IN, Operational Agents were provided with an optimal algorithm due to [McMahon & Florian 75] that minimises the maximum lateness of the jobs assigned to them. For the sake of completeness, the algorithm is presented in the appendix A.

The difference between EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN lies in the inclusion (EXPLICIT-OAS-IN) or not of the Operational Agents. In EXPLICIT-OAS-IN, OAs were assigned an algorithm that minimises the maximum lateness. Other algorithms can be assigned to the OAs, depending on the particular problem and objectives to be achieved. Considering the high modularity of the system, it is very easy to provide the Operational Agents with a different algorithm. The choice of the algorithm that minimises the maximum lateness as an optimisation criterion for the OAs took into consideration the fact that one of the main objectives adopted for the experiments was to meet the due-dates. The utility function assigned to the RTAs, for both versions (see

[2]Operational Agents of different RTAs can play different roles.

chapter 5, section 5.3.4), took into consideration the same main objective. However, it is important to point out that, despite the fact that the RTAs and OAs are specialised in meeting the due-dates, this does not mean it was the only objective provided to the whole system. It just means that the system was customised to perform better in that area. However, other aspects were also embodied in the system, particularly in terms of optimising the utilisation of resources.

The description of the cases used to test EXPLICIT is presented in the next section.

9.2 A Battery of Cases

In order to test the performance of EXPLICIT, a battery of 54 cases was generated. In this section the generator of cases is described. The main features of the 54 cases are listed.

9.2.1 The Generator of Cases

The parameters adopted for the generation of the data for test purposes are discussed in this section.

The pattern of the Job

The generation of the data followed three pre-defined "patterns":

- Tightly Linked Operations - this "pattern" is "very tight" in the sense that the order of the operations is completely defined a priori. Figure 9.1 represents a job whose operations are "tightly" linked. This "pattern" is identified by "Type A".

- Semi-Tightly Linked Operations - this "pattern" introduces some degrees of freedom to the way the operations can be linked, though it is still "tight". In this pattern each operation must follow one of two operations and must be followed by one of two other operations. Figure 9.2 represents a job whose operations are

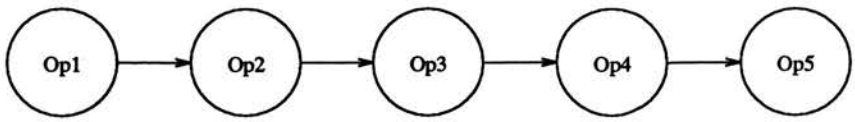

Figure 9.1: Job with tightly linked operations - Type A

"semi-tightly" linked. This "pattern" is identified by "Type B".

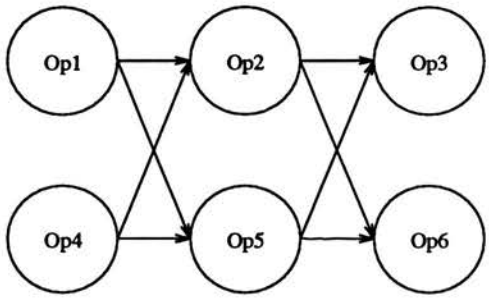

Figure 9.2: Job with semi-tightly linked operations - Type B

- Loosely Linked Operations - The third pattern considered is very "loose". In this pattern only the first and last operations of the job are defined a priori. The rest of the operations do not have a predefined order. Figure 9.3 represents a job whose operations are "loosely" linked. This "pattern" is identified by "Type C".

Class	*Number of cases*	*Lower limit*	*Upper limit*
1	6	5	10
2	6	10	20
3	3	20	25
4	3	25	30

Table 9.1: The limits of the uniform distribution for the number of jobs per case

Other parameters of the generator of cases

- Number of Jobs - the number of jobs for each case follows a uniform distribution. The lower and upper limit of the uniform distribution are given as input. For

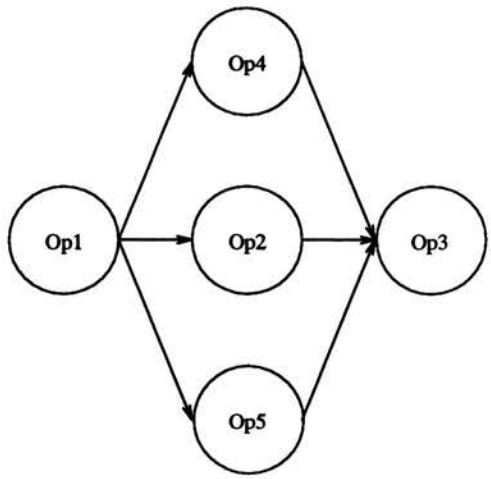

Figure 9.3: Job with loosely linked operations - Type C

the three types of jobs, four classes were considered, as shown in table 9.1. For instance, for each type of job, for class 1, 6 cases were generated. For each case the number of jobs follows a uniform distribution with an upper and lower limit of 5 and 10, respectively.

- Available time of each job - the available time of each job follows a uniform distribution. The lower and upper limit for the uniform distribution are given as input. The values adopted as lower and upper limit of the uniform distribution are 480 and 540[3], respectively.

- Due-date of each job - the due-date is calculated by applying a certain rate to the earliest finish time of the job. The rate to apply to the earliest finish time of the job follows a uniform distribution. The values adopted as lower and upper limit of the uniform distribution are 1.25 and 1.4 respectively.

- Operations - Six different operations are given as input.

  - dai - The Daily Telegraph
  - eur - The European
  - fin - The Financial Times

[3]The unit is minutes. 480 = 60 minutes * 8 hours; 540 = 60 minutes * 9 hours. The idea is to have jobs arriving at the shop between 8 a.m. and 9 a.m.. However the unit is not relevant.

- gua - The Guardian
- ind - The Independent
- sco - The Scotsman

- Number of operations per job - all the jobs have all the operations.
- Duration of operations - the duration of each operation follows a uniform distribution. For each operation, the lower and upper limit of the uniform distribution are given as input. Table 9.2 shows the values adopted as lower and upper limits of the uniform distribution[4] for each operation.

Operation	*Lower limit*	*Upper limit*
dai	5	16
eur	5	16
fin	10	91
gua	10	41
ind	10	41
sco	5	16

Table 9.2: The limits of the uniform distribution for the duration of operations

- Number of machines per aggregate resource - three cases are considered:
  - 2 machines per aggregate resource for classes 1 and 2
  - 3 machines per aggregate resource for class 3
  - 5 machines per aggregate resource for class 4

In appendix C the main features of the cases used to test the performance of the system are summarised.

9.3 Performance of the System

In this section the versions EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN are compared with each other with respect to a set of performance measures. Considering the same

[4]The implicit unit is again minutes, but as mentioned above, the unit is not relevant.

set of performance measures, the results produced by EXPLICIT-OAS-IN are compared with the results produced by four dispatch rules.

9.3.1 Performance Measures

It is not easy to define objectives for scheduling problems nor is it easy to evaluate alternative schedules. Selecting a set of performance measures to evaluate solutions is not a simple task, considering that there are numerous criteria, and they are complex and often conflicting.

Let us introduce some notation:

- S - the set of jobs $J_1, J_2, \ldots J_n$ to be scheduled
- n - the total number of jobs to be scheduled
- M - the set of aggregate resources $M_1, M_2, \ldots M_m$
- m - the total number of aggregate resources
- im - the total number of individual machines
- d_i - the due-date of job i, i.e., the promised delivery date of job i
- r_i - the ready time of job i, i.e., the time the job i is available for processing
- C_i - the completion time of job i, i.e., the time at which the processing of job i finishes
- F_i - the flow time of job i, i.e., the time that job i spends in the workshop. Thus:

$$F_i = C_i - r_i$$

- L_i - the lateness of job i. The difference between its completion time and its due-date:

$$L_i = C_i - d_i$$

Note that when the job is early, i.e., when it completes before its due date, L_i is

negative. It is often more useful to have a variable which, unlike lateness, only takes non-zero values when a job is *tardy*, i.e., when it completes after its due-date.

- T_i - the tardiness of job i

$$T_i = max\{L_i, 0\}$$

- I_{jik} - the idle time of machine j between two consecutive jobs, job i and job k, is the time that elapses between the completion of job i and the start of job k

- I_j - the total idle time of machine j. Let us consider the set of m jobs performed on machine m_j and ordered by ascending start times, $J_{(1)}, J_{(2)}, \ldots, J_{(m)}$. The total idle time is given by:

$$I_j = \sum_{i=1}^{(m-1)} I_{ji(i+1)}$$

The set of performance measures selected is:

- From the jobs perspective:
  - $MaxCT$ - Maximum Completion Time over all the jobs, also called the *total production time* or the *make-span*, i.e.,

$$MaxCT = max_{i \in S} C_i$$

  - $MeanCT$- Mean Completion Time over all the jobs, i.e.,

$$MeanCT = 1/n \sum_{i=1}^{n} C_i$$

  - $MaxFT$ - Maximum Flow Time over all the jobs, i.e.,

$$MaxFT = max_{i \in S} F_i$$

  - $MaxLat$ - Maximum Lateness over all the jobs, i.e.,

$$MaxLat = max_{i \in S} L_i$$

  - $MaxTard$ - Maximum Tardiness over all the jobs, i.e.,

$$MaxTard = max_{i \in S} T_i$$

– *TardyJobs* - Number of Tardy Jobs, i.e.,

$$TardyJobs = \sum_{i=1}^{n} x_i$$

where:

$$x_i = \begin{cases} 1 & \text{if } T_i > 0 \\ 0 & otherwise \end{cases}$$

- From the resources perspective:

  – *TotIT* - Total Idle Time for all machines i.e.,

$$TotIT = \sum_{j=1}^{im} I_j$$

  – *MeanITAR* - Mean Idle Time per Aggregate Resource, i.e.,

$$MeanITAR = \frac{TotIT}{m}$$

  – *MeanITM* - Mean Idle Time per Machine, i.e.,

$$MeanITM = \frac{TotIT}{im}$$

The set of performance measures selected for testing purposes is far from being exhaustive. It would be difficult, even impossible, to be exhaustive. As a compromise, there was an attempt to select a set of measures that seem to have some consensus in the AI and OR literature (see e.g., [Panwalkar & Iskander 77], [French 82], [Blackstone *et al* 82], [Park *et al* 84], [Kurtulus & Narula 85]).

9.3.2 With or Without Operational Agents?

In this section the performance of the two versions, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, is compared considering the set of performance measures defined in the previous section.

Appendix D includes detailed results for EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, with respect to the performance measures defined in the previous section.

Tardiness

In terms of tardiness, there seems to be a consensus that the most important measure is the "Number of Tardy Jobs" and, as a secondary measure, the "Maximum Tardiness".

Figure 9.4[5] displays the comparison between EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT with respect to "Number of Tardy Jobs". For each case the reduction in the number of tardy jobs induced by EXPLICIT-OAS-IN was divided by the number of jobs. The idea was to provide a relative measure to compare the different cases, rather than just the absolute reduction in the number of tardy jobs. For example, for the case number 10, EXPLICIT-OAS-IN generates 6 tardy jobs while EXPLICIT-OAS-OUT generates 10 tardy jobs. The number of jobs in the case 10 is 20. So, the relative reduction in the number of tardy jobs induced by EXPLICIT-OAS-IN (per job) is $4/20 = 0.20$ for the case 10.

The graph shows that EXPLICIT-OAS-IN clearly outperforms EXPLICIT-OAS-OUT. The number of cases where EXPLICIT-OAS-OUT performs better than EXPLICIT-OAS-IN is 6 while the number of cases where EXPLICIT-OAS-IN performs better than EXPLICIT-OAS-OUT is 14. In terms of the different types of jobs, the improvement of EXPLICIT-OAS-IN over EXPLICIT-OAS-OUT is particularly marked for jobs type A and C (cases 1 to 18 and 37 to 54). However, even for type B jobs, EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-OUT.

Figure 9.5 displays the comparison between EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT with respect to "Maximum Tardiness". For each case the reduction in the maximum tardiness induced by EXPLICIT-OAS-IN was divided by the maximum completion time ($MaxCT$) induced by EXPLICIT-OAS-OUT. Again, the idea was to provide a relative measure to compare the different cases, rather than just the absolute reduction in the maximum tardiness. The duration of the longest job ($MaxCT$) according to EXPLICIT-OAS-OUT was used as reference. For example, for the case number 11, the maximum tardiness generated by EXPLICIT-OAS-IN is 100 while EXPLICIT-OAS-OUT generated a

[5]On all the comparison graphs the following criterion is adopted: for the cases where the performance of the first compared system is better than the performance of the second compared system, a positive value is displayed; a negative value corresponds to the situations where the second compared system performs better than the first one.

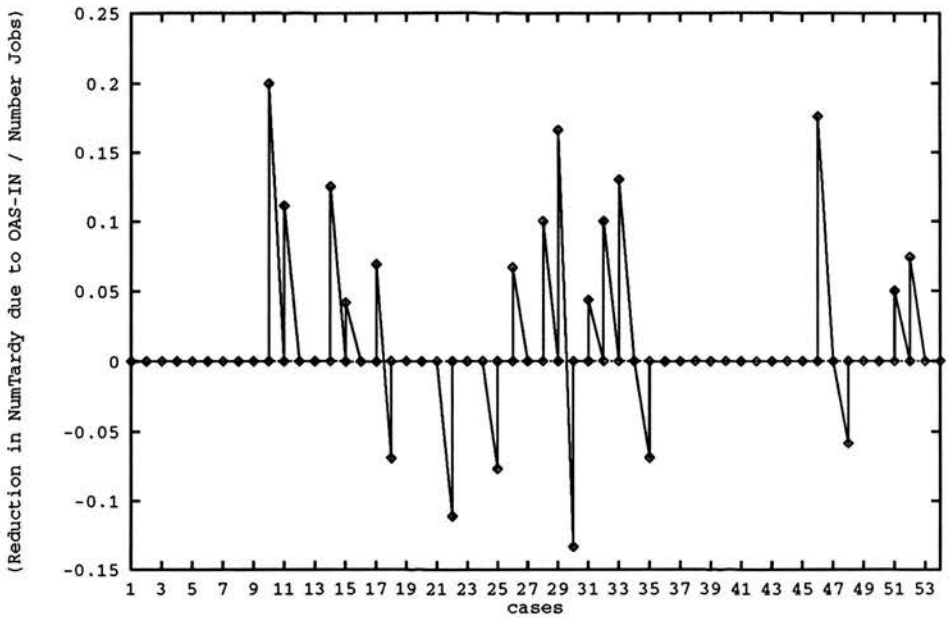

Figure 9.4: Relative reduction in NumTardy(EXPLICIT-OAS-INvs.EXPLICIT-OAS-OUT)

maximum tardiness of 134. The maximum completion time (according to EXPLICIT-OAS-OUT) for the case 11 is 1081. So, the relative reduction in the maximum tardiness induced by EXPLICIT-OAS-IN (relative to $MaxCT$) is $(134 - 100)/1081 = 0.031$ for the case 11.

The graph shows that EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-IN, though in a few cases, EXPLICIT-OAS-OUT performs considerably better than EXPLICIT-OAS-IN. The number of cases where EXPLICIT-OAS-OUT performs better than EXPLICIT-OAS-IN is 7 while the number of cases where EXPLICIT-OAS-IN performs better than EXPLICIT-OAS-OUT is 15.

Recall that the difference between the two versions is that EXPLICIT-OAS-IN is provided with Operational Agents whose expertise is to minimise the maximum lateness.

Let us define:

- N - the number of cases
- T_{out-i} - the number of tardy jobs generated by EXPLICIT-OAS-OUT, for the case i

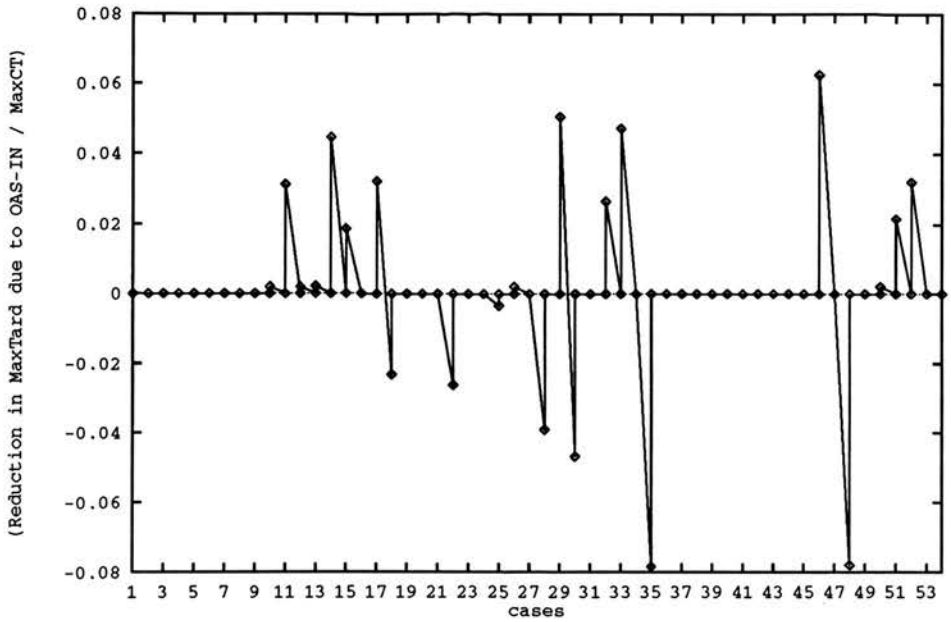

Figure 9.5: Relative reduction in MaxTard(EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT)

- T_{in-i} - the number of tardy jobs generated by EXPLICIT-OAS-IN, for the case i
- $RedTard_{in,out-i}$ - the reduction in the number of tardy jobs when applying EXPLICIT-OAS-IN in comparison to the number of tardy jobs generated by EXPLICIT-OAS-OUT, for the case i

$$RedTard_{in,out-i} = \begin{cases} T_{out-i} - T_{in-i} & \text{if } T_{out-i} > T_{in-i} \\ 0 & \text{if } T_{out-i} \leq T_{in-i} \end{cases}$$

- $RedTard_{out,in-i}$ - the reduction in the number of tardy jobs when applying EXPLICIT-OAS-OUT in comparison to the number of tardy jobs generated by EXPLICIT-OAS-IN, for the case i

$$RedTard_{out,in-i} = \begin{cases} T_{in-i} - T_{out-i} & \text{if } T_{in-i} > T_{out-i} \\ 0 & \text{if } T_{in-i} \leq T_{out-i} \end{cases}$$

- MT_{out-i} - the maximum tardiness for the case i when applying EXPLICIT-OAS-OUT
- MT_{in-i} - the maximum tardiness for the case i when applying EXPLICIT-OAS-IN

- $RedMaxT_{in,out-i}$ - the reduction in the maximum tardiness due to EXPLICIT-OAS-IN in comparison to the maximum tardiness generated by EXPLICIT-OAS-OUT, for the case i

$$RedMaxT_{in,out-i} = \begin{cases} MT_{out-i} - MT_{in-i} & \text{if } MT_{out-i} > MT_{in-i} \\ 0 & \text{if } MT_{out-i} \leq MT_{in-i} \end{cases}$$

- $RedMaxT_{out,in-i}$ - the reduction in the maximum tardiness due to EXPLICIT-OAS-OUT in comparison to the maximum tardiness due to EXPLICIT-OAS-IN, for the case i

$$RedMaxT_{out,in-i} = \begin{cases} MT_{in-i} - MT_{out-i} & \text{if } MT_{in-i} > MT_{out-i} \\ 0 & \text{if } MT_{in-i} \leq MT_{out-i} \end{cases}$$

Considering the above definitions, two aggregate ratios can be defined:

- $RPRTardyJobs_{in,out}$ - relative power of reduction in the number of tardy jobs of EXPLICIT-OAS-IN compared to EXPLICIT-OAS-OUT

$$RPRTardyJobs_{in,out} = \frac{\sum_{i=1}^{N} RedTard_{in,out-i}}{\sum_{i=1}^{N} RedTard_{out,in-i}}$$

- $RPRMaxTard_{in,out}$ - relative power of reduction in the maximum tardiness of EXPLICIT-OAS-IN compared to EXPLICIT-OAS-OUT

$$RPRMaxTard_{in,out} = \frac{\sum_{i=1}^{N} RedMaxT_{in,out-i}}{\sum_{i=1}^{N} RedMaxT_{out,in-i}}$$

The ratio $RPRTardyJobs_{in,out}$ compares the reduction in the number of tardy jobs due to EXPLICIT-OAS-IN (compared to the number of tardy jobs generated by EXPLICIT-OAS-OUT) with the reduction in the number of tardy jobs due to EXPLICIT-OAS-OUT (compared to the number of tardy jobs generated by EXPLICIT-OAS-IN). The greater the ratio, the better the performance that EXPLICIT-OAS-IN exhibits compared to EXPLICIT-OAS-OUT.

The ratio $RPRMaxTard_{in,out}$ compares the reduction in the maximum tardiness generated by EXPLICIT-OAS-IN (compared to the maximum tardiness generated by

EXPLICIT-OAS-OUT) with the reduction in the maximum tardiness due to EXPLICIT-OAS-OUT (compared to the maximum tardiness generated by EXPLICIT-OAS-IN). The greater the ratio, the better the performance that EXPLICIT-OAS-IN exhibits compared to EXPLICIT-OAS-OUT.

Aggregate Ratio	*Value*
$RPRTardyJobs_{in,out}$	3.33
$RPRMaxTard_{in,out}$	1.38

Table 9.3: Aggregate ratios for Tardiness - EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT

Table 9.3 shows the values for the two ratios. Clearly, EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-OUT in terms of tardiness measures. The reduction in the number of tardy jobs due to EXPLICIT-OAS-IN (compared to EXPLICIT-OAS-OUT) is 3.33 the reduction in the number of tardy jobs due to EXPLICIT-OAS-OUT (compared to EXPLICIT-OAS-IN). In terms of maximum tardiness, the reduction in the maximum tardiness due to EXPLICIT-OAS-IN (compared to EXPLICIT-OAS-OUT) is 1.38 the reduction in the maximum tardiness due to EXPLICIT-OAS-OUT (compared to EXPLICIT-OAS-IN).

The difference between the two models, EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, is the inclusion of Operational Agents in the model EXPLICIT-OAS-IN. The main expertise of the Operational Agents is to manage tardiness. So, it is not a surprise that the model EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-OUT in terms of tardiness.

Completion Times

In terms of completion times, EXPLICIT-OAS-IN performs slightly better than EXPLICIT-OAS-OUT, though the improvement from EXPLICIT-OAS-OUT to EXPLICIT-OAS-IN is much smaller than what was observed in terms of tardiness.

Figures 9.6, 9.7 and 9.8 display the comparison between the two versions of EXPLICIT with respect to the performance measures $MeanCT$, $MaxCT$ and $MaxFT$. In each graph it is displayed the percentage reduction due to EXPLICIT-OAS-IN (compared to EXPLICIT-OAS-OUT) with respect to the considered measure. For instance, considering

the measure $MeanCT$ (figure 9.6), the case number 29, EXPLICIT-OAS-IN generates a solution with $MeanCT = 792.39$ while EXPLICIT-OAS-OUT generates a solution with $MeanCT = 851.00$. So, the percentage reduction in $MeanCT$ due to EXPLICIT-OAS-IN and relatively to EXPLICIT-OAS-OUT is $(1 - 792.39/851.00) * 100) = 6.89\%$

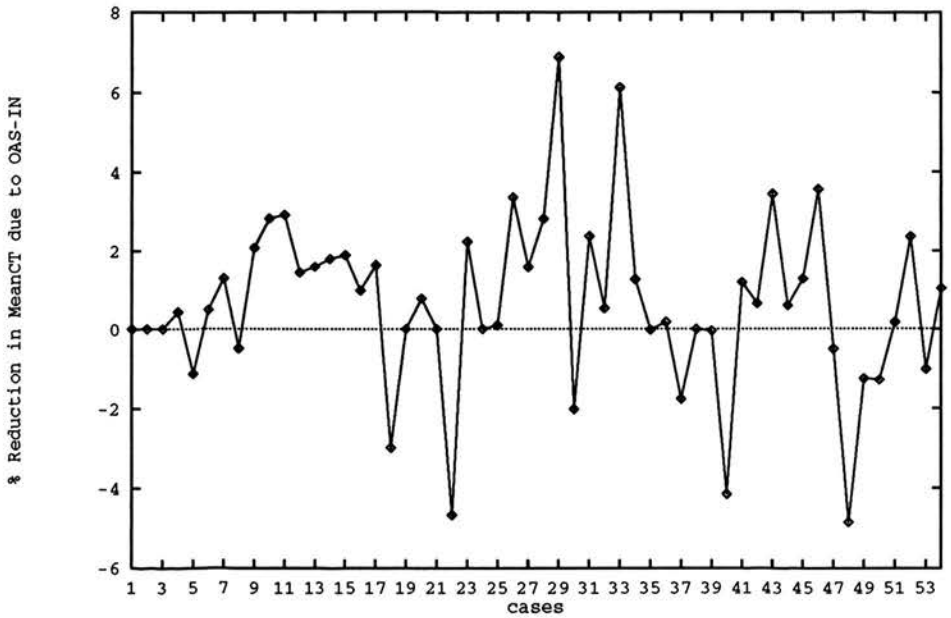

Figure 9.6: Percentage Reduction in MeanCT (EXPLICIT-OAS-OUT vs. EXPLICIT-OAS-IN)

In order to globally compare the results produced by the two versions in terms of these performance measures, and analogously to the aggregate ratios defined for tardiness[6], the following aggregate ratio was defined.

Let us define:

- N - the number of cases
- $measureX_iA$ - measure X for the case i when applying the approach A
- $measureX_iB$ - measure X for the case i when applying the approach B

[6]In the case of tardiness, since $NumTardy$ and $MaxTard$ can have value 0, different aggregate ratios had to be defined

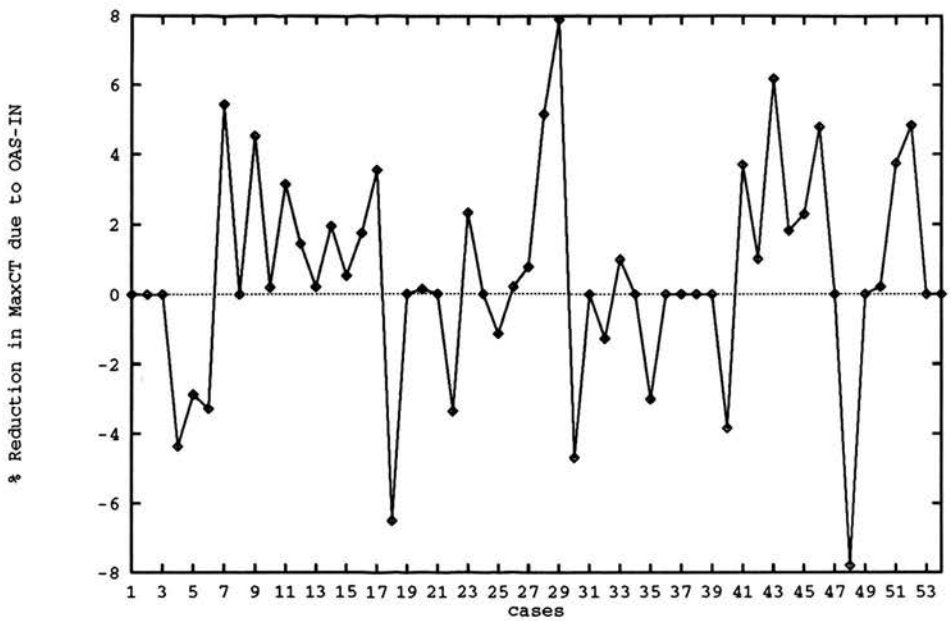

Figure 9.7: Percentage Reduction in MaxCT(EXPLICIT-OAS-OUTvs.EXPLICIT-OAS-IN)

- $AvRmeasureX_{A,B}$ - average percentage reduction of measure X due to heuristic A compared to heuristic B.

The aggregate ratio is defined by:

$$AvRmeasureX_{A,B} = 100/N\sum_{i=1}^{N}(1 - \frac{measureX_iA}{measureX_iB})$$

The aggregate ratio $AvRmeasureX_{AB}$ compares the performance of heuristic A with the performance of heuristic B, in terms of measure X. It is an average over all the cases. The aggregate ratio gives the percentage improvement of heuristic A over heuristic B. As an example let us consider the measure $Meanct$. If the ratio is, say 5%, it means than on average heuristic A induces a reduction of 5% in $Meanct$ and comparatively to heuristic B.

For instance, $AvRMeanCT_{in,out}$ compares the reduction due to EXPLICIT-OAS-IN, relatively to EXPLICIT-OAS-OUT, in terms of $MeanCT$, considering the average per case.

Table 9.4 shows the values for the different aggregate ratios comparing EXPLICIT-OAS-IN with EXPLICIT-OAS-OUT with respect to the performance measures $MeanCT$,

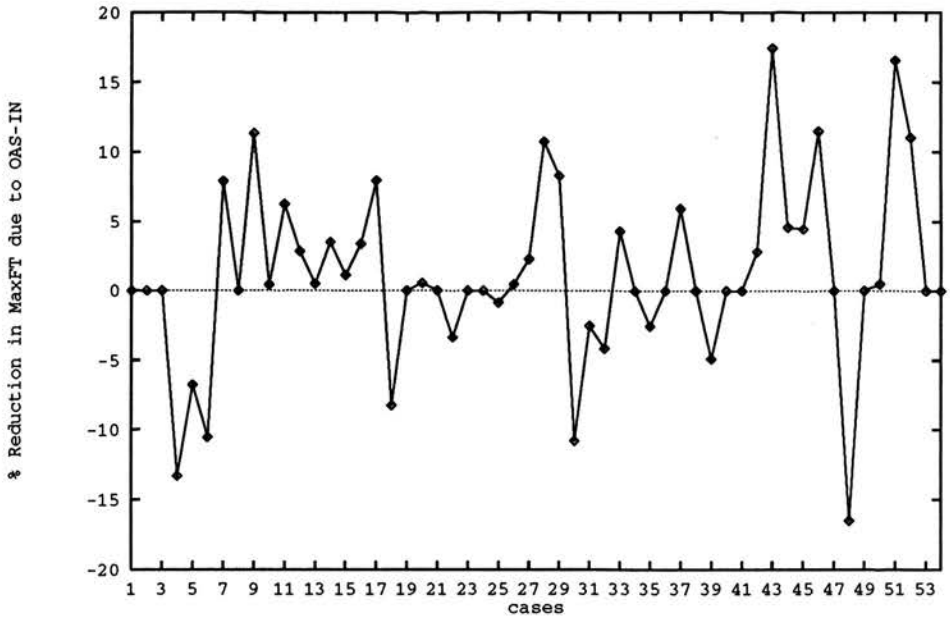

Figure 9.8: Percentage Reduction in MaxFT(EXPLICIT-OAS-OUTvs.EXPLICIT-OAS-IN)

$MaxCT$ and $MaxFT$. EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-OUT, though, the average improvement of EXPLICIT-OAS-IN over EXPLICIT-OAS-OUT, per case, in what concerns completion times is smaller than what was observed in terms of tardiness, 1% for $MeanCT$ and $MaxCT$ and 2% for $MaxFT$

Aggregate Ratio	*Value*
$AvRMeanCT_{in,out}$	1%
$AvRMaxCT_{in,out}$	1%
$AvRMaxFT_{in,out}$	2%

Table 9.4: Aggregate ratios for Completion Times - EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT

Utilisation of Resources

In terms of utilisation of resources, again EXPLICIT-OAS-IN performs significantly better than EXPLICIT-OAS-OUT.

Figure 9.9 and table 9.5 show the comparison between the two versions of EXPLICIT with

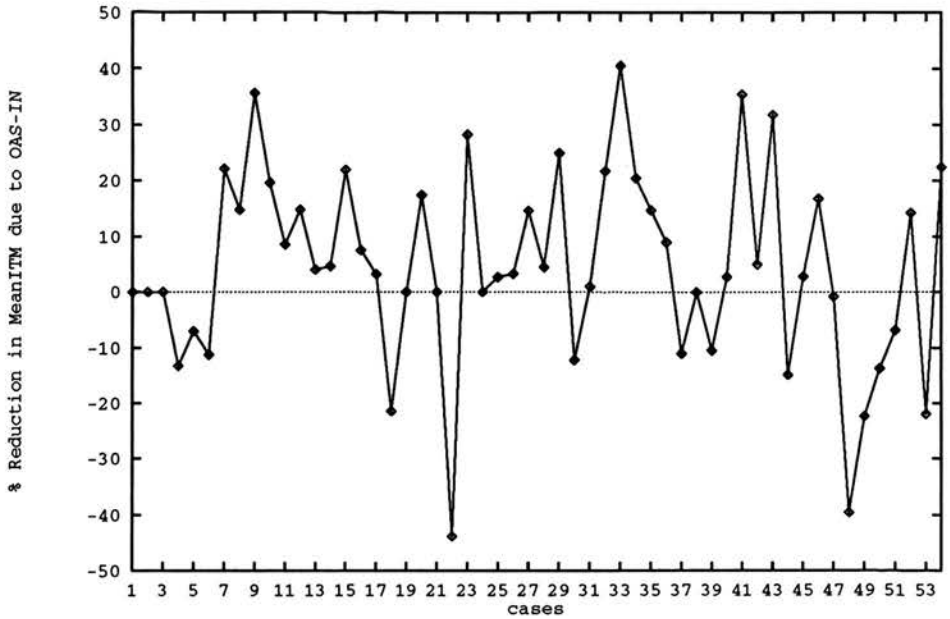

Figure 9.9: Percentage Reduction in MeanITM (EXPLICIT-OAS-OUT vs. EXPLICIT-OAS-IN)

Aggregate Ratio	*Value*
$AvRMeanITM_{in,out}$	5%

Table 9.5: Aggregate ratios for Mean Idle Time per machine - EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT

respect to the performance measure $MeanITM$. The value of $AvRMeanITM_{in,out}$ is 5% which means that, on average, EXPLICIT-OAS-IN compared to EXPLICIT-OAS-OUT induces a reduction of the mean idle time per machine of about 5%.

EXPLICIT-OAS-OUT vs. EXPLICIT-OAS-IN

In this section the performance of the two versions of EXPLICIT was compared. The difference between the two versions lies in the exclusion of the Operational Agents, in the case of the version EXPLICIT-OAS-OUT, while EXPLICIT-OAS-IN is provided with Operational Agents. Operational Agents are specialised in minimising the maximum lateness of the jobs assigned to their machines and, implicitly, to guarantee a better

utilisation of their machines.

Clearly, EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-OUT in all the performance measures considered, but especially in the areas in which Operational Agents are specialised, i.e., meeting the due-dates and optimising the utilisation of machines. Table 9.6 summarises the values for the aggregate ratios comparing EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT. Because of its superiority, the performance of EXPLICIT-OAS-IN is compared with the performance of four popular dispatch rules in the next section.

Aggregate Ratio	*Value*
$RPRTardyJobs_{in,out}$	3.33
$RPRMaxTard_{in,out}$	1.38
$AvRMeanCT_{in,out}$	1%
$AvRMaxCT_{in,out}$	1%
$AvRMaxFT_{in,out}$	2%
$AvRMeanITM_{in,out}$	5%

Table 9.6: Aggregate ratios for EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT

9.3.3 EXPLICIT vs. Dispatch Rules

In this section the performance of EXPLICIT-OAS-IN is compared with the performance of four popular dispatch rules, considering the battery of cases presented in section 9.2 and the measures defined in terms of tardiness, completion times and utilisation of resources in section 9.3.1.

The Dispatch Rules

The sequencing rules against which EXPLICIT-OAS-IN was tested are four popular priority dispatch rules. The comments that were made with respect to the selection of a set of performance measures apply to the selection of the set of dispatch rules as well. It would be difficult, even impossible, to be exhaustive. As a compromise, there was an attempt to select a set of dispatch rules that have been reported as particularly good at the two major (sub)objectives that are embodied in EXPLICIT — meeting of due-dates and optimising the utilisation of resources (see e.g.,

[Gere 66], [Kan 76], [Panwalkar & Iskander 77], [Blackstone *et al* 82], [Park *et al* 84], [Kurtulus & Narula 85])).

The definitions of the four dispatch rules considered are given below:

- SOF - Shortest Operation First

$$\text{SOF} = \text{Min } d_{ij},$$

Tie-breaker: FCFS[7].

Where:

d_{ij} is the duration of the j^{th} operation of the i^{th} job.

- MOF - Maximum Operation First

$$\text{MOF} = \text{Max } d_{ij}$$

Tie-breaker: FCFS

- MINSLK - Minimum Slack First

$$\text{MINSLK} = \text{Min } slk_{ij}$$

Tie-breaker: FCFS

Where:

$slk_{ij} = lst_{ij} - Max(est_{ij}, time)$ and terms are the same as the ones used for the Job Tactical Agent, defined in standard PERT/CPM[8]

- MAXSLK - Maximum Slack First

$$\text{MAXSLK} = \text{Max } slk_{ij}$$

Tie-breaker: FCFS

The order of assignment of the operations followed the order defined by the dispatch rules listed above. The assignment of a machine to an operation was done always considering the first available machine.

[7] FCFS - First Come First Served. The first eligible activity is assigned the highest priority. Tie-breaker: random

[8] See e.g., [Willis 85], [Lockyer 84], [Davis & Patterson 75] and [Fendley 68].

9.3.4 Comparison of EXPLICIT with the Dispatch Rules

Due to the clear superiority of EXPLICIT-OAS-IN over EXPLICIT-OAS-OUT, this version was considered for the comparison with the selected dispatch rules. From now on EXPLICIT-OAS-IN will be referred to by simply EXPLICIT.

This section presents the results of the comparison of EXPLICIT with SOF, MOF, MINSLK and MAXSLK successively for tardiness, completion times and utilisation of resources. The graphical results and the aggregate ratios will be presented to compare the performance of EXPLICIT with each of the dispatch rules, in a similar way as it was done for the comparison between EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT.

Tardiness

The comparison of EXPLICIT with the four dispatch rules in terms of "Number of Tardy Jobs" is shown in figures 9.10, 9.11, 9.12 and 9.13.

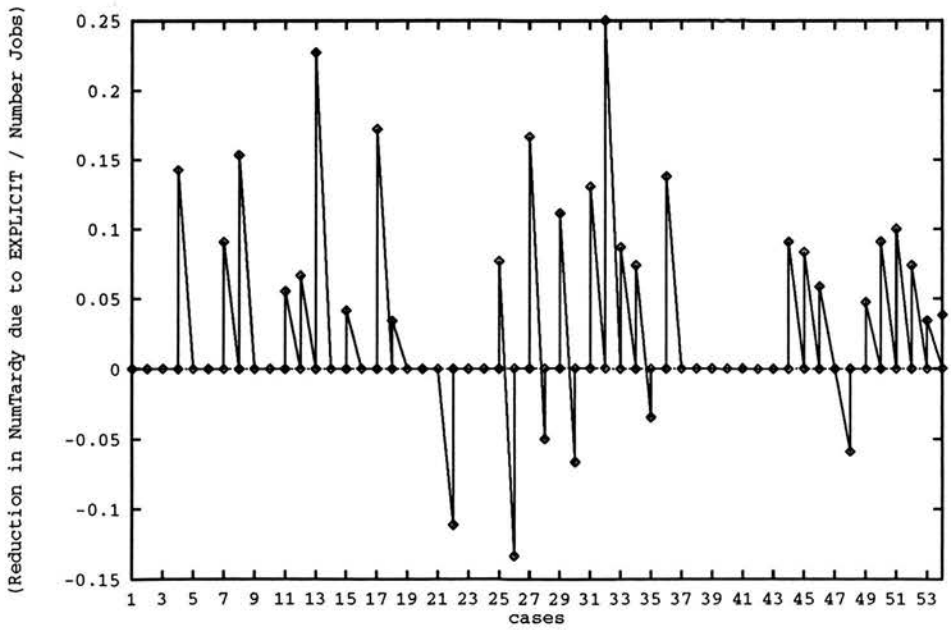

Figure 9.10: Relative reduction in NumTardy (EXPLICIT vs. SOF)

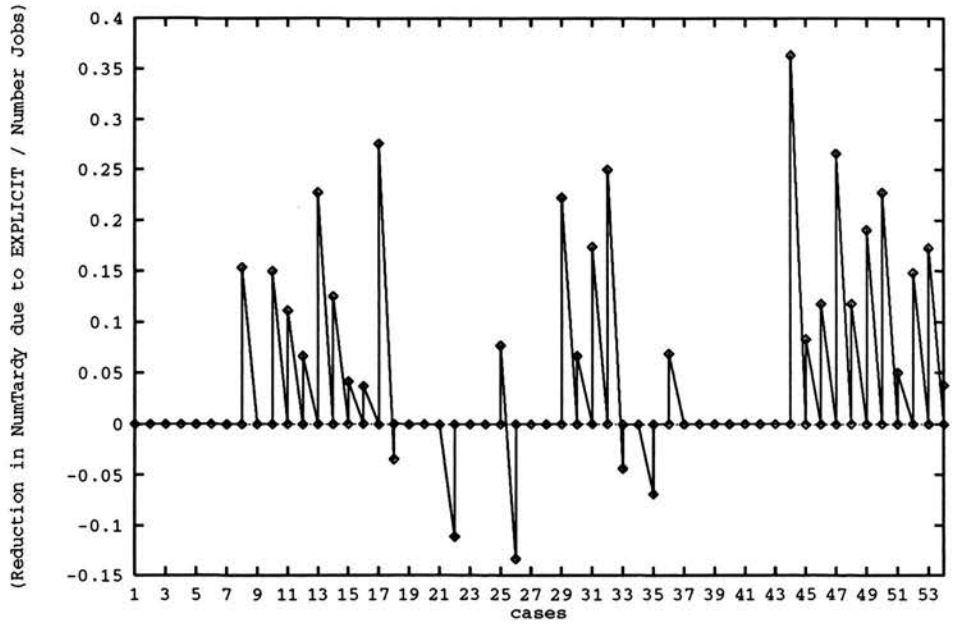

Figure 9.11: Relative reduction in NumTardy (EXPLICIT vs. MOF)

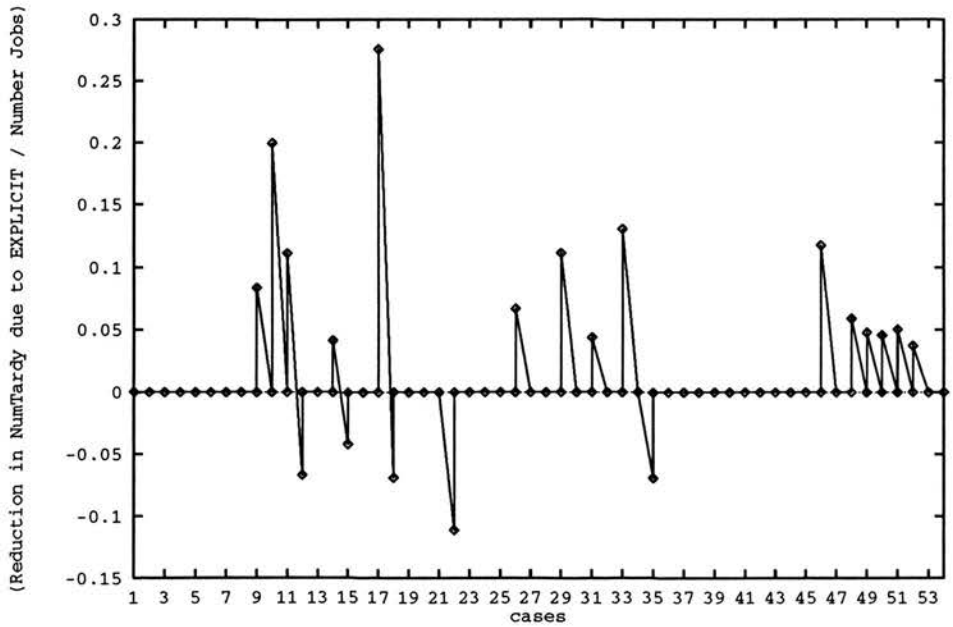

Figure 9.12: Relative reduction in NumTardy (EXPLICIT vs. MINSLK)

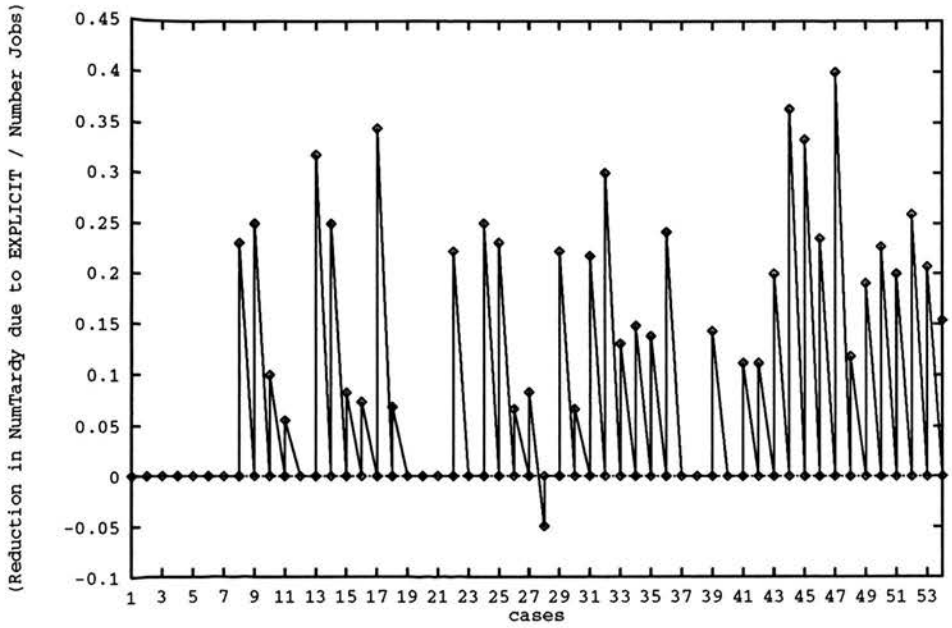

Figure 9.13: Relative reduction in NumTardy (EXPLICIT vs. MAXSLK)

From the analysis of the graphs, it is clear that EXPLICIT outperforms the four dispatch rules in terms of $NumTardy$. Recall that the graph displays, for each case, the reduction in the number of tardy jobs due to EXPLICIT divided by the number of jobs. The dispatch rule MAXSLK performs particularly bad. MAXSLK reduces the number of tardy jobs only in one case, while EXPLICIT reduces the number of tardy jobs in 38 cases, and the magnitude of the reduction in the number of tardy jobs due to EXPLICIT is very significant for each case. From the four dispatch rules, MINSLK is the dispatch rule that performs better, though clearly worse than EXPLICIT. The number of cases for which EXPLICIT reduces the number of tardy jobs is 15, while MINSLK only reduces the number of tardy jobs in 5 cases. Additionally, the magnitude of reduction in the number of tardy jobs due to EXPLICIT for each case is greater than the reduction in the number of tardy jobs due to MINSLK, in particular for the case 17. In terms of type of jobs, it is difficult to differentiate the behaviour of EXPLICIT against the four dispatch rules for a particular type of jobs.

Analogously to what was done for the comparison between EXPLICIT-OAS-IN with EXPLICIT-OAS-OUT, aggregate ratios were calculated for the "Number of Tardy Jobs". Table 9.7 displays the aggregate ratios $RPRTardyJobs_{explicit,X}$, i.e., the relative power of reduction in the number of tardy jobs of EXPLICIT compared to each dispatch rule. Each aggregate ratio was calculated comparing EXPLICIT with one dispatch rule.

The conclusions reported above are reflected in the ratios. MAXSLK performs terribly, EXPLICIT's power of reduction in the number of tardy jobs is 142.85 times the power of reduction in the number of tardy jobs of MAXSLK; MINSLK is the dispatch rule that performs better, though EXPLICIT's power of reduction in the number of tardy jobs is still 4.35 times the power of reduction in the number of tardy jobs of MINSLK.

Figures 9.14, 9.15, 9.16 and 9.17 display the comparison between EXPLICIT and the four dispatch rules in terms of "Maximum Tardiness".

From the analysis of the graphs concerning the maximum tardiness, again it is clear that EXPLICIT performs better than the dispatch rules. The outperformance of EXPLICIT is even more noticeable in terms of "Maximum Tardiness" than is terms of

Aggregate Ratio	*Value*
$RPRTardyJobs_{explicit,sof}$	7.14
$RPRTardyJobs_{explicit,mof}$	11.11
$RPRTardyJobs_{explicit,minslk}$	4.35
$RPRTardyJobs_{explicit,maxslk}$	142.85

Table 9.7: Aggregate ratios for Number of Tardy Jobs - EXPLICIT vs. SOF, MOF, MINSLK and MAXSLK

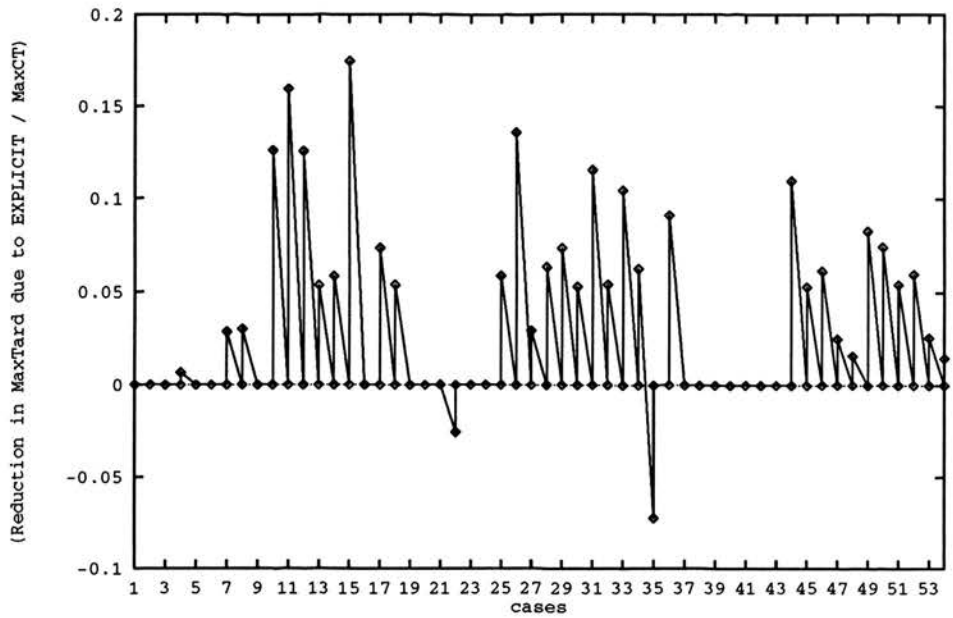

Figure 9.14: Relative reduction in MaxTard (EXPLICIT vs. SOF)

"Number of Tardy Jobs". The cases 28 and 35 are the notable exceptions. For this case, MINSLK and MOF perform much better than EXPLICIT. The comments that were made with respect to the comparison of the different dispatch rules in terms of "Number of Tardy Jobs" also apply to "Maximum Tardiness". MAXSLK performs terribly. MINSLK is the dispatch rule that performs better compared to EXPLICIT, though EXPLICIT outperforms MINSLK.

Table 9.8 shows the aggregate ratios $RPRMaxTard_{explicit,X}$, i.e., the relative power of reduction in the maximum tardiness of EXPLICIT compared to each dispatch rule. Each aggregate ratio was calculated comparing EXPLICIT with one dispatch rule. The

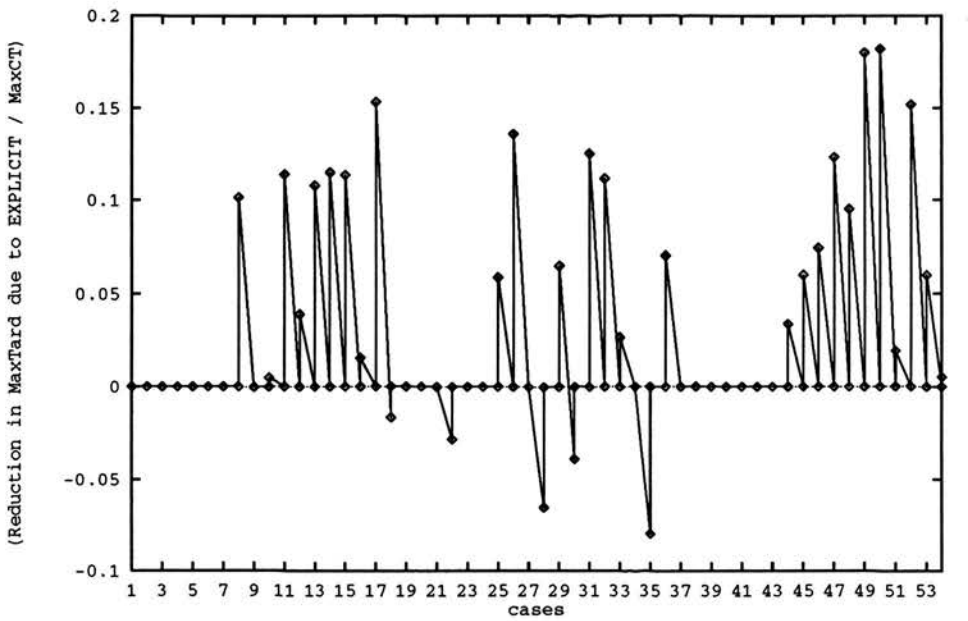

Figure 9.15: Relative reduction in MaxTard (EXPLICIT vs. MOF)

conclusions reported above are reflected in the ratios - MAXSLK performs terribly, EXPLICIT's power of reduction of the maximum tardiness is 333.33 times the power of reduction in the maximum tardiness of MAXSLK; MINSLK is the dispatch rule that performs better, though EXPLICIT's power of reduction of the maximum tardiness is still 2.43 times the power of reduction in the maximum tardiness of MINSLK. The corresponding values when comparing EXPLICIT with MOF and SOF are respectively 33.33 and 11.11.

Aggregate Ratio	*Value*
$RPRMaxTard_{explicit,sof}$	33.33
$RPRMaxTard_{explicit,mof}$	11.11
$RPRMaxTard_{explicit,minslk}$	2.43
$RPRMaxTard_{explicit,maxslk}$	333.33

Table 9.8: Aggregate ratios for Maximum Tardiness - EXPLICIT-OAS-IN VS. SOF, MOF, MINSLK and MAXSLK

As a conclusion, it is remarkable the outperformance of EXPLICIT over the four dispatch rules in terms of tardiness.

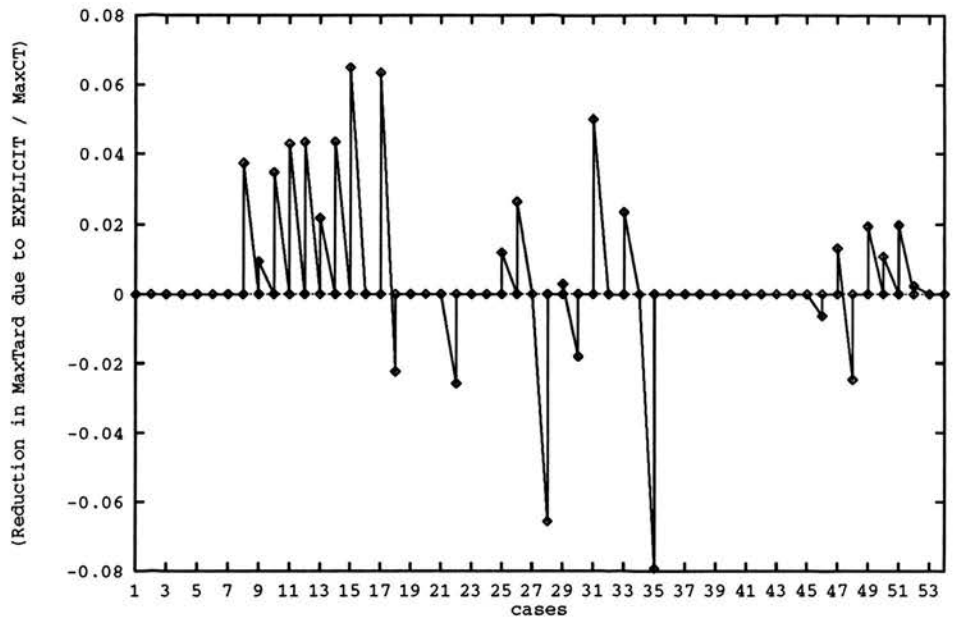

Figure 9.16: Relative reduction in MaxTard (EXPLICIT vs. MINSLK)

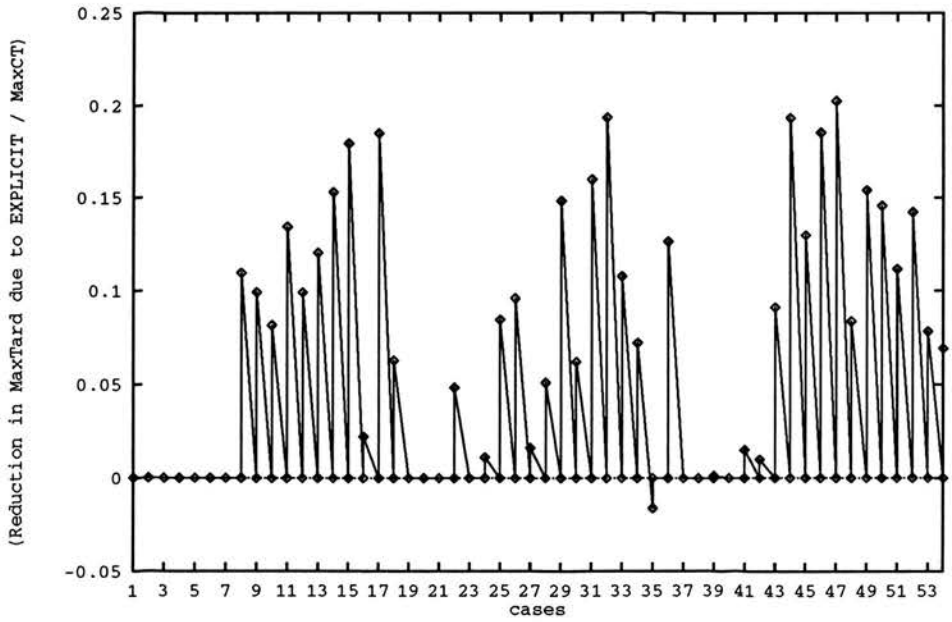

Figure 9.17: Relative reduction in MaxTard (EXPLICIT vs. MAXSLK)

Completion Times

Figures 9.18, 9.19, 9.20, 9.21 display the comparison between EXPLICIT and each of the four dispatch rules in terms of "Mean Completion Time". Figures 9.22, 9.23, 9.24, 9.25 display the comparison between EXPLICIT and each of the four dispatch rules in terms of "Maximum Completion Time". Figures 9.26, 9.27, 9.28, 9.29 display the comparison between EXPLICIT and each of the four dispatch rules in terms of "Maximum Flow Time". Tables 9.9, 9.10 and 9.11 show the values for the aggregate ratios comparing EXPLICIT with each of the priority dispatch rules for the completion times measures $MaxCT$, $MeanCT$ and $MaxFT$.

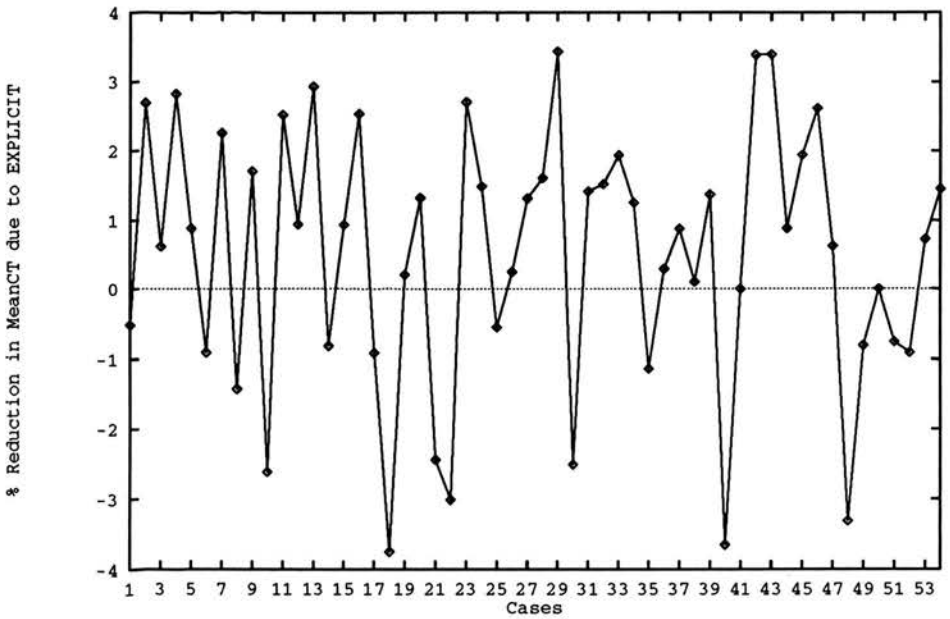

Figure 9.18: Percentage Reduction in MeanCT (EXPLICIT vs. SOF)

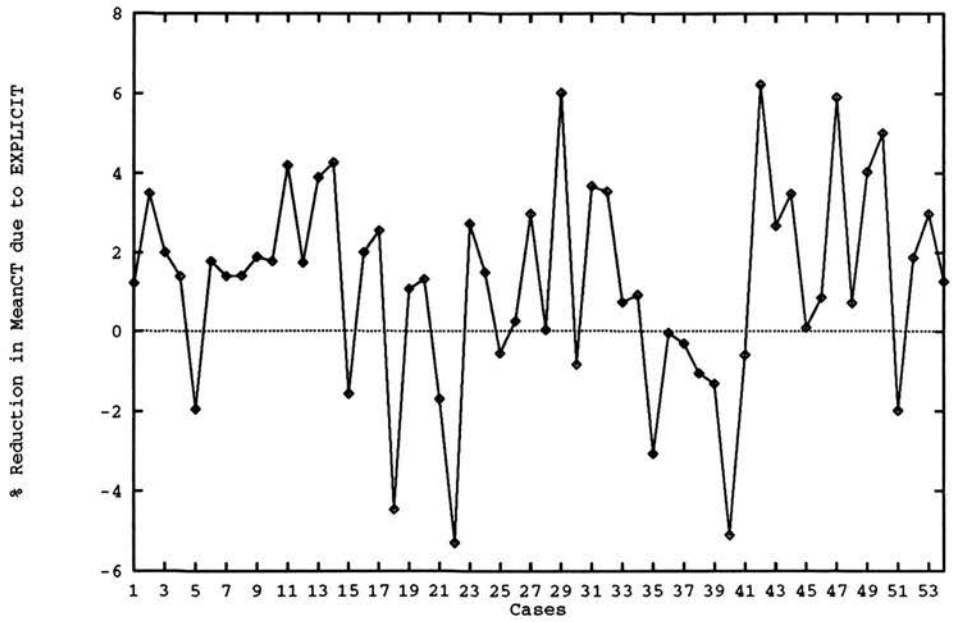

Figure 9.19: Percentage Reduction in MeanCT (EXPLICIT vs. MOF)

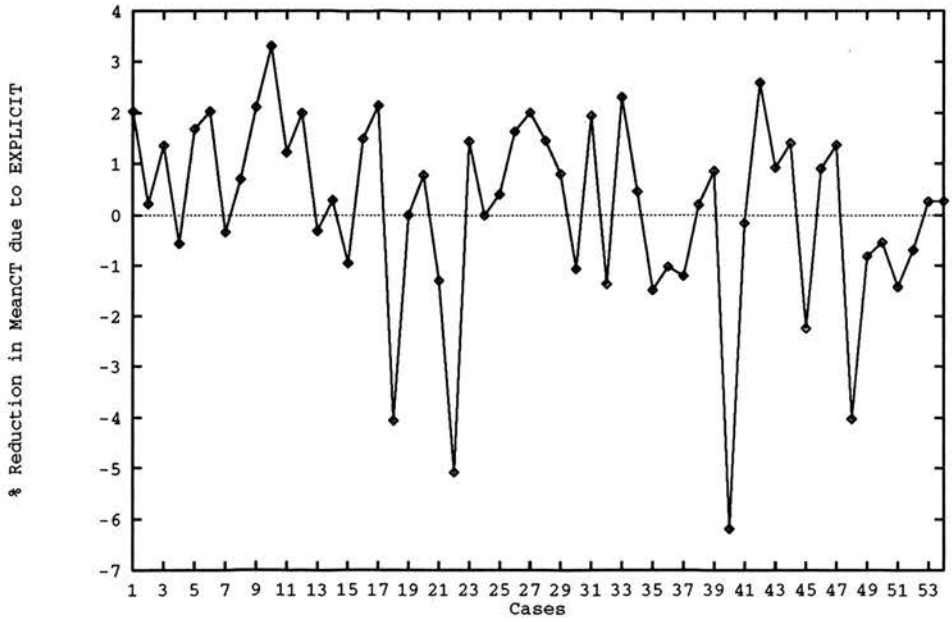

Figure 9.20: Percentage Reduction in MeanCT (EXPLICIT vs. MINSLK)

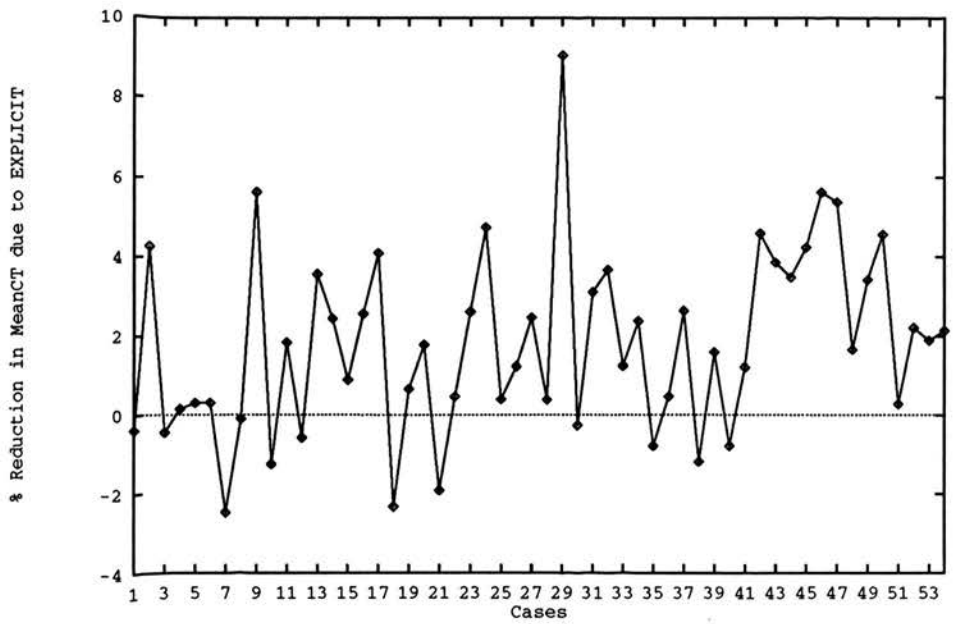

Figure 9.21: Percentage Reduction in MeanCT (EXPLICIT vs. MAXSLK)

In terms of mean completion time ($MeanCT$), from the analysis of the figures 9.18, 9.19, 9.20, 9.21 it is difficult to conclude which heuristic performs better. EXPLICIT seems to perform better than MAXSLK and MOF, less clear when comparing its performance with SOF and MINSLK. Especially in this case, EXPLICIT seems to perform better than MINSLK on average, but in 4 cases EXPLICIT performs considerable worse than MINSLK. From the analysis of the table 9.9 one can conclude that EXPLICIT induces a reduction of 1%-2%, on average, compared to the other approaches, in terms of $MeanCT$.

Aggregate Ratio	*Value*
$AvRMeanCT_{explicit,sof}$	1%
$AvRMeanCT_{explicit,mof}$	2%
$AvRMeanCT_{explicit,minslk}$	1%
$AvRMeanCT_{explicit,maxslk}$	2%

Table 9.9: Aggregate ratios for Mean Completion Time - EXPLICIT-OAS-IN VS. SOF, MOF, MINSLK and MAXSLK

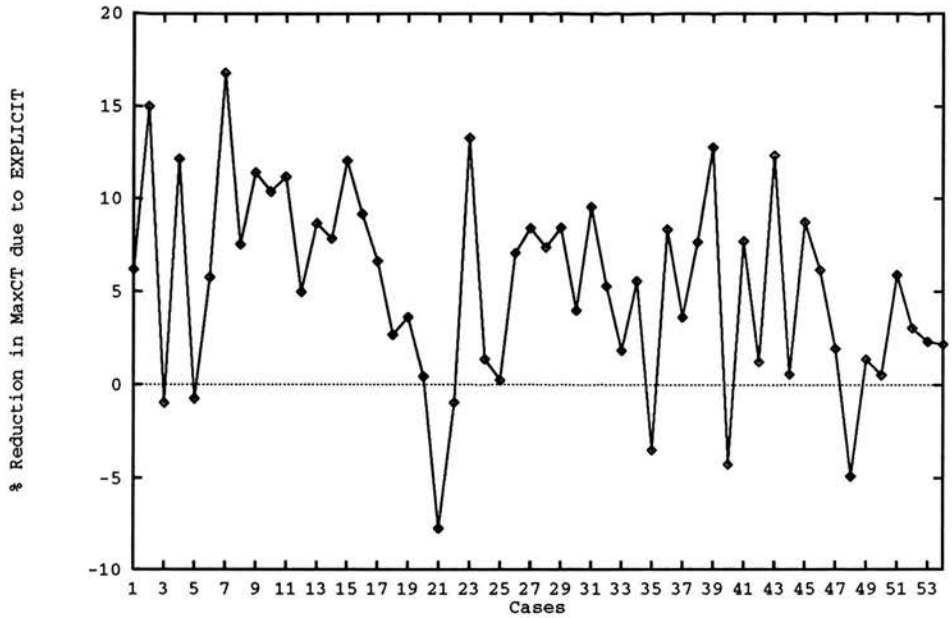

Figure 9.22: Percentage Reduction in MaxCT (EXPLICIT vs. SOF)

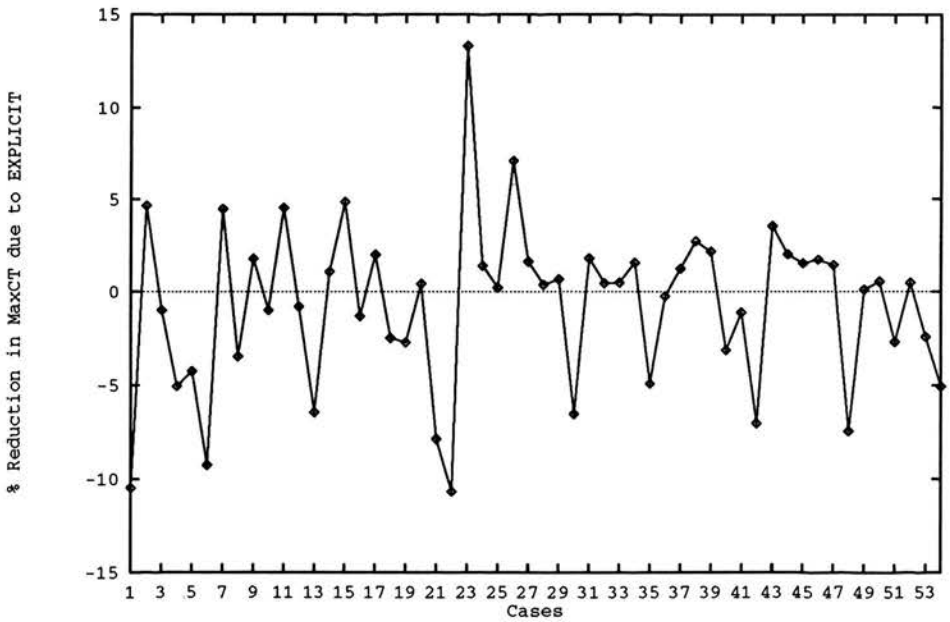

Figure 9.23: Percentage Reduction in MaxCT (EXPLICIT vs. MOF)

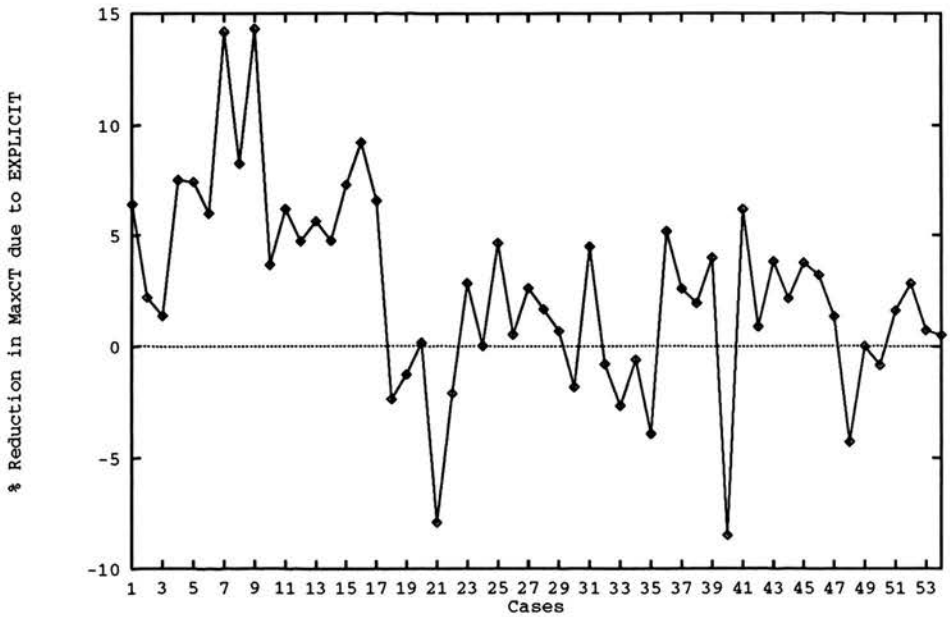

Figure 9.24: Percentage Reduction in MaxCT (EXPLICIT vs. MINSLK)

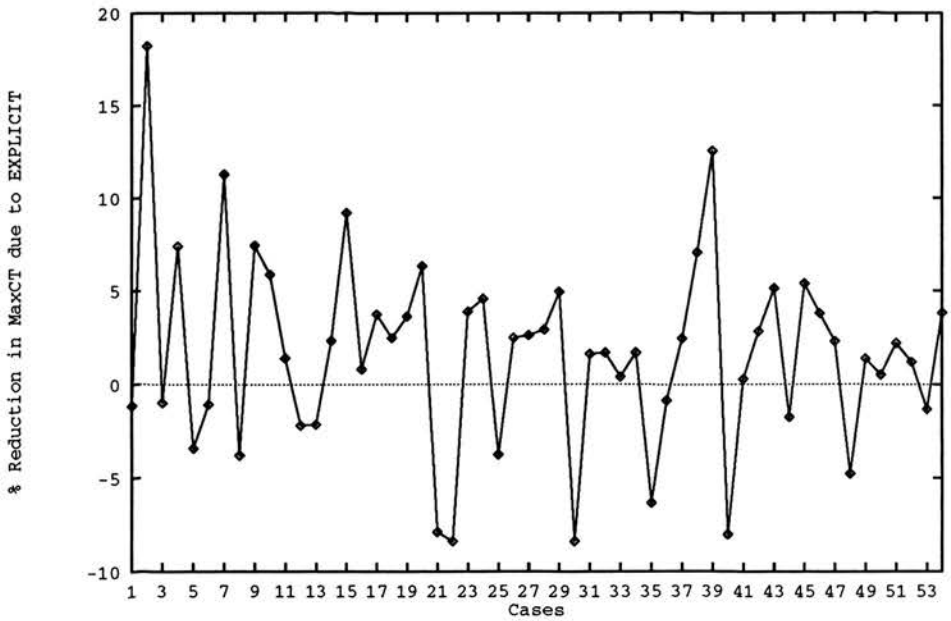

Figure 9.25: Percentage Reduction in MaxCT (EXPLICIT vs. MAXSLK)

In terms of maximum completion time ($MaxCT$), from the analysis of the figures 9.22, 9.23, 9.24, 9.25, it is clear that EXPLICIT performs better than MINSLK, MAXSLK and SOF, though there are some cases where EXPLICIT performs worse than the dispatch rules. Between EXPLICIT and MOF it is hard to tell which one performs better, since their behaviour alternate from case to case. From the analysis of the table 9.10 one can verify that, on average, the priority dispatch rule MOF performs similarly to EXPLICIT in terms of $MaxCT$, the total production time. Recall that the ratios for the completion times show the average reduction due to EXPLICIT with respect to a given measure. Since the value of $MaxCT_{explicit,mof}$ is 0.00, that means the two approaches have a similar behaviour, on average, and in what concerns the $MaxCT$. Comparing EXPLICIT with MINSLK, the aggregate ratio $MaxCT_{explicit,minslk}$ shows that, on average, EXPLICIT generates a reduction of 3% of the total production time generated by the dispatch rule MINSLK. This value is 2% when comparing EXPLICIT with MAXSLK and 6% when comparing EXPLICIT with SOF.

Aggregate Ratio	*Value*
$AvRMaxCT_{explicit,sof}$	6%
$AvRMaxCT_{explicit,mof}$	0%
$AvRMaxCT_{explicit,minslk}$	3%
$AvRMaxCT_{explicit,maxslk}$	2%

Table 9.10: Aggregate ratios for Maximum Completion Time (Make Span) - EXPLICIT-OAS-IN vs. SOF, MOF, MINSLK and MAXSLK

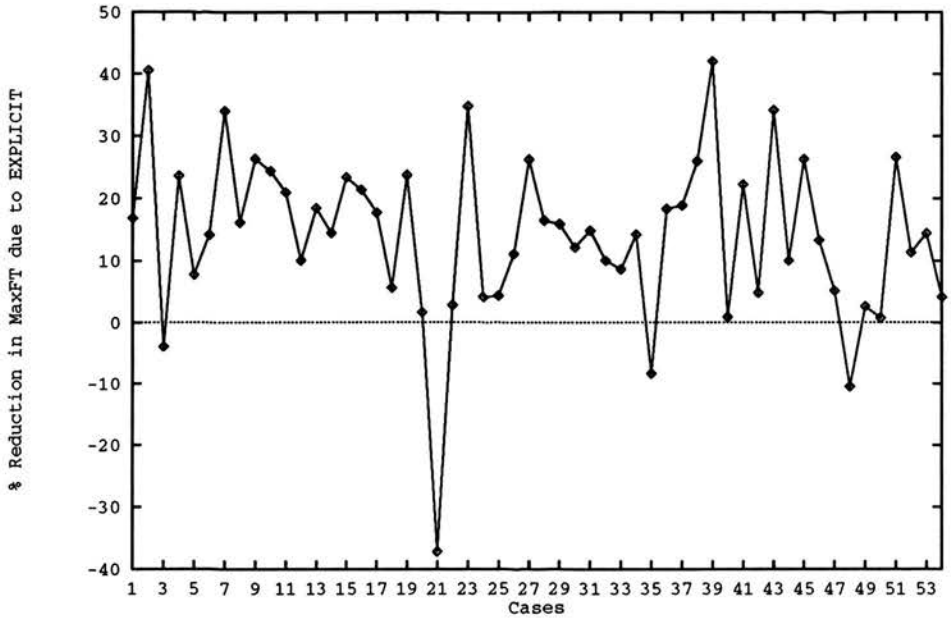

Figure 9.26: Percentage Reduction in MaxFT (EXPLICIT vs. SOF)

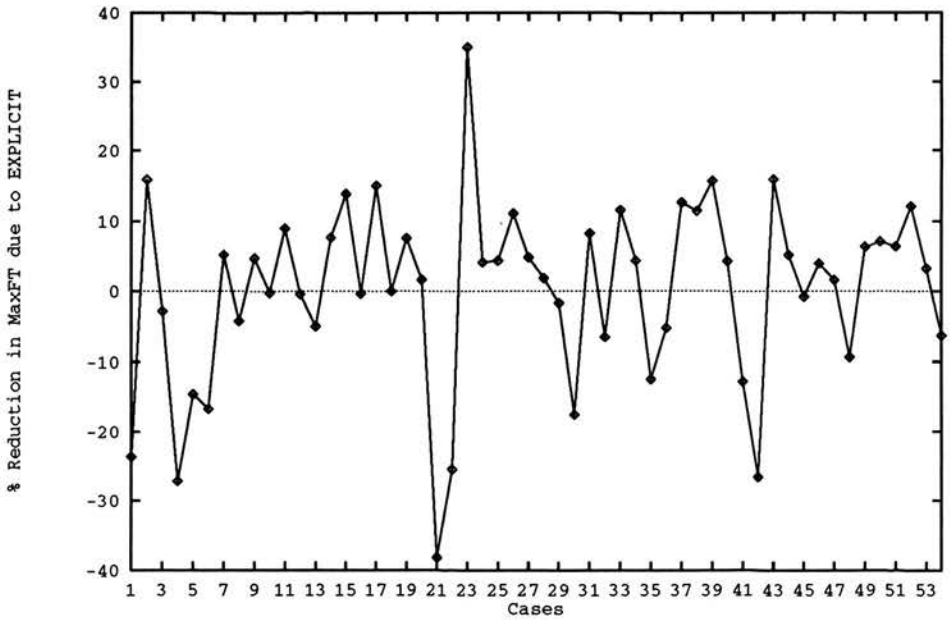

Figure 9.27: Percentage Reduction in MaxFT (EXPLICIT vs. MOF)

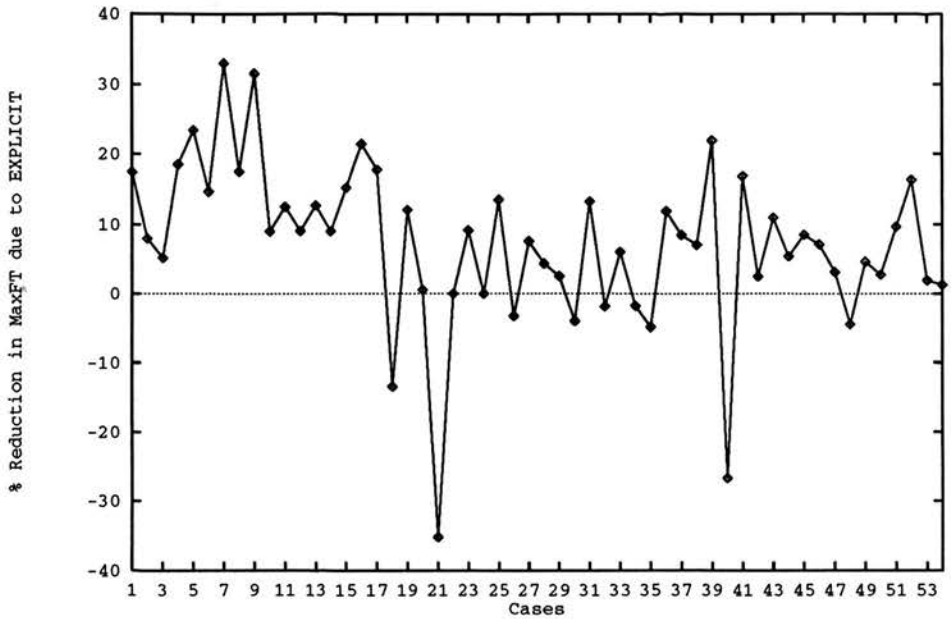

Figure 9.28: Percentage Reduction in MaxFT (EXPLICIT vs. MINSLK)

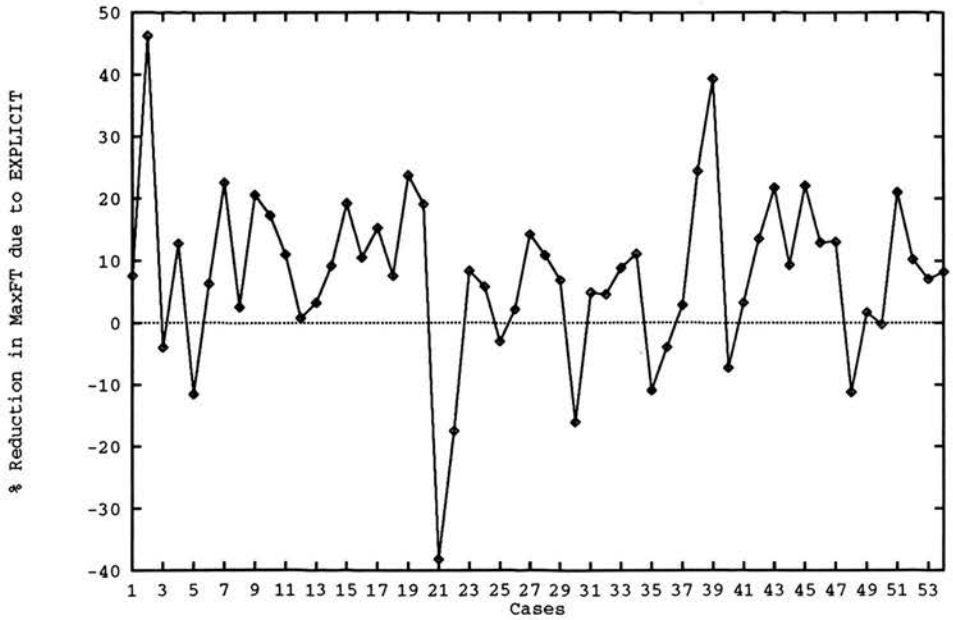

Figure 9.29: Percentage Reduction in MaxFT (EXPLICIT vs. MAXSLK)

In terms of maximum flow time ($MaxFT$), from the analysis of the figures 9.26, 9.27, 9.28, 9.29 it is it is noticeable that EXPLICIT slightly outperforms the four dispatch rules. The case 21 is the notable exception. For this case, every dispatch rule performs better than EXPLICIT. Comparing EXPLICIT with MOF, though EXPLICIT seems to perform better than MOF there are a few cases where EXPLICIT performs considerably worse than MOF. From the analysis of the table 9.11 one can conclude that EXPLICIT clearly outperforms the dispatch rules SOF, MAXSLK and MINSLK, generating a reduction in terms of the maximum flow time, on average, of 14%, 9% and 8% respectively. The behaviour of EXPLICIT is similar to the behaviour of MOF in what concerns the maximum flow time. On average, EXPLICIT brings about a reduction of 1% of the maximum flow time, in comparison with the rule MOF.

Aggregate Ratio	*Value*
$AvRMaxFT_{explicit,sof}$	14%
$AvRMaxFT_{explicit,mof}$	1%
$AvRMaxFT_{explicit,minslk}$	8%
$AvRMaxFT_{explicit,maxslk}$	8%

Table 9.11: Aggregate ratios for Maximum Flow Time - EXPLICIT-OAS-IN vs. SOF, MOF, MINSLK and MAXSLK

As a conclusion, in terms of completion times, EXPLICIT clearly performs better than SOF, MINSLK and MAXSLK. The dispatch rule MOF performs similarly to EXPLICIT, though EXPLICIT slightly outperforms MOF.

Utilisation of Resources

Figures 9.30, 9.31, 9.32, and 9.33 display the performance of EXPLICIT in comparison to the four dispatch rules, SOF, MOF, MINSLK and MAXSLK in terms of the mean idle time per machine ($MeanITM$). From the observation of the graphs, EXPLICIT performs better than MINSLK, MAXSLK and SOF, though there are some notable exceptions. For instance for the cases 21, 40, and 48, EXPLICIT performs worse than any of the dispatch rules. The heuristic MOF clearly outperforms EXPLICIT, especially for the type C jobs.

Table 9.12 displays the values of the aggregate ratio $AvRMeanITM$. From the analysis of these values, one can infer that, on average, EXPLICIT performs better that SOF, MINSLK and MAXSLK in terms of the mean idle time per machine. On average, EXPLICIT brings about a reduction of the mean idle time per machine of 15%, 6% and 14%, when comparing it with SOF, MINSLK and MAXSLK respectively. MOF clearly outperforms EXPLICIT. On average, EXPLICIT generates an increase of the mean idle time per machine of about 13% in comparison with MOF.

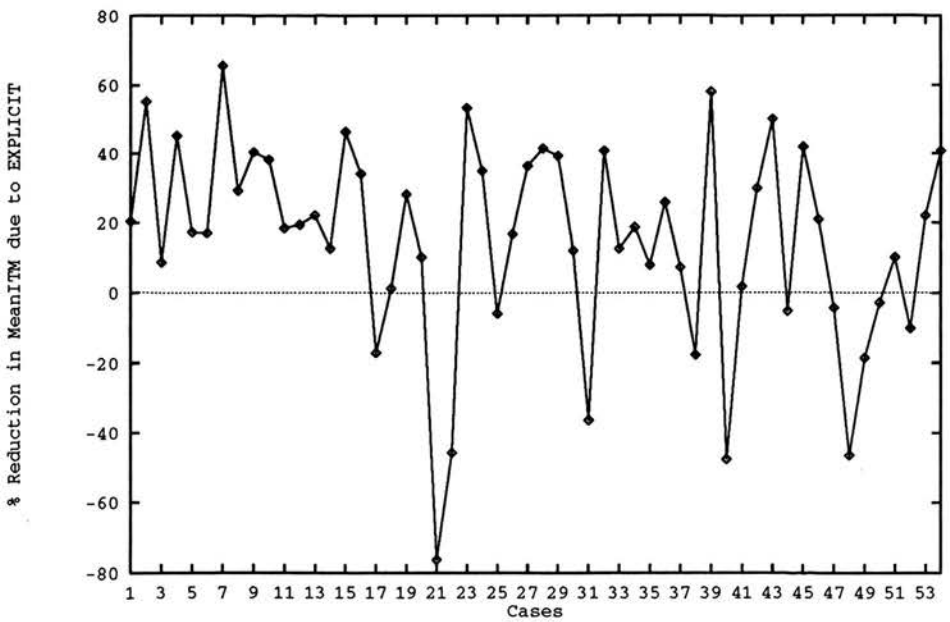

Figure 9.30: Percentage Reduction in MeanITM (EXPLICIT vs. SOF)

Table 9.13 displays the values of the different aggregate ratios for EXPLICIT and the

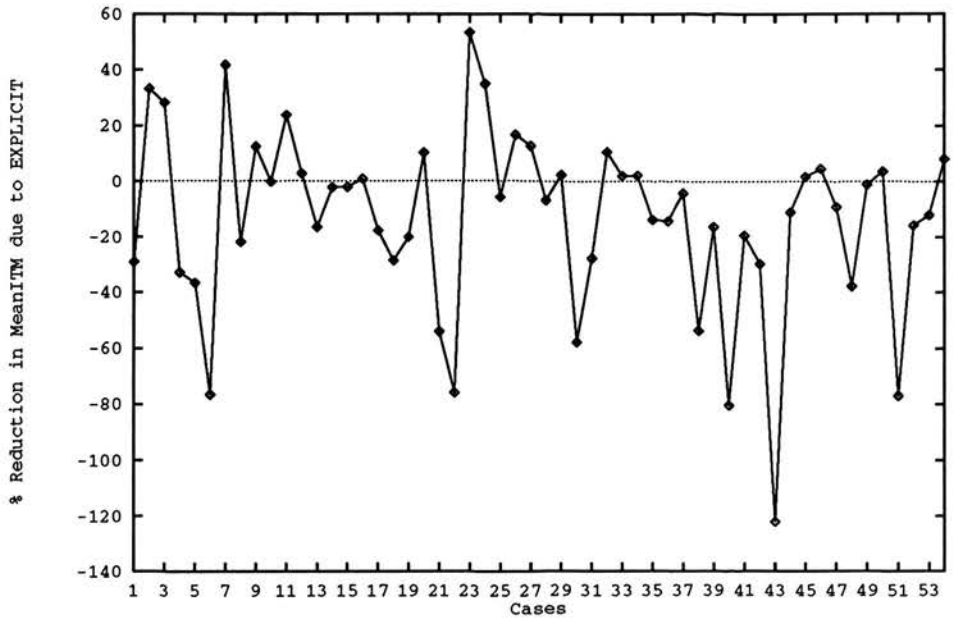

Figure 9.31: Percentage Reduction in MeanITM (EXPLICIT vs. MOF)

Aggregate Ratio	*Value*
$AvRMeanITM_{explicit,sof}$	15%
$AvRMeanITM_{explicit,mof}$	-13%
$AvRMeanITM_{explicit,minslk}$	6%
$AvRMeanITM_{explicit,maxslk}$	14%

Table 9.12: Aggregate ratios for Mean Idle Time per Machine - EXPLICIT-OAS-IN vs. SOF, MOF, MINSLK and MAXSLK

four dispatch rules. As conclusions in terms of the comparison between EXPLICIT and the dispatch rules SOF, MOF, MINSLK and MAXSLK, one can conclude:

- EXPLICIT outperforms the four dispatch rules in terms of tardiness. Among the four dispatch rules, MINSLK is the one that performs better in terms of tardiness. MOF performs very poorly, compared to EXPLICIT.
- In terms of completion times, EXPLICIT clearly performs better than SOF, MINSLK and MAXSLK. The dispatch rule MOF performs similarly to EXPLICIT, though EXPLICIT slightly outperforms MOF.

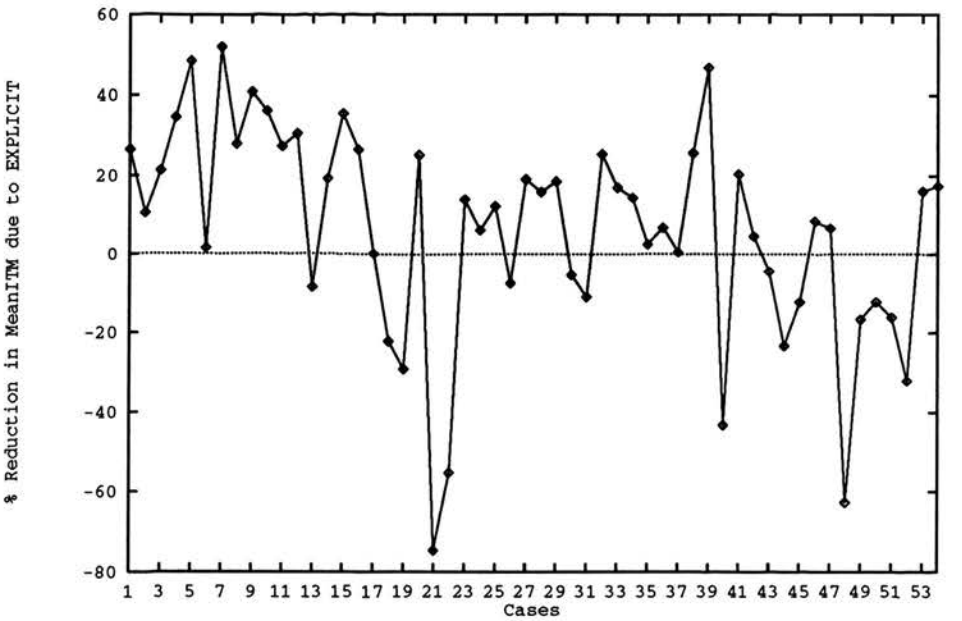

Figure 9.32: Percentage Reduction in MeanITM (EXPLICIT vs. MINSLK)

Aggregate Ratio	SOF	MOF	MINSLK	MAXSLK
$RPRTardyJobs_{explicit,X}$	7.14	11.11	4.35	142.85
$RPRMaxTard_{explicit,X}$	33.33	11.11	2.43	333.33
$AvRMeanCT_{explicit,X}$	1%	2%	1%	2%
$AvRMaxCT_{explicit,X}$	6%	0%	3%	2%
$AvRMaxFT_{explicit,X}$	14%	1%	8%	8%
$AvRMeanITM_{explicit,X}$	15%	-13%	6%	14%

Table 9.13: Aggregate ratios for EXPLICIT-OAS-IN vs. SOF, MOF, MINSLK and MAXSLK

- In terms of utilisation of resources, EXPLICIT performs better than SOF, MINSLK and MAXSLK. The dispatch rule MOF outperforms EXPLICIT.

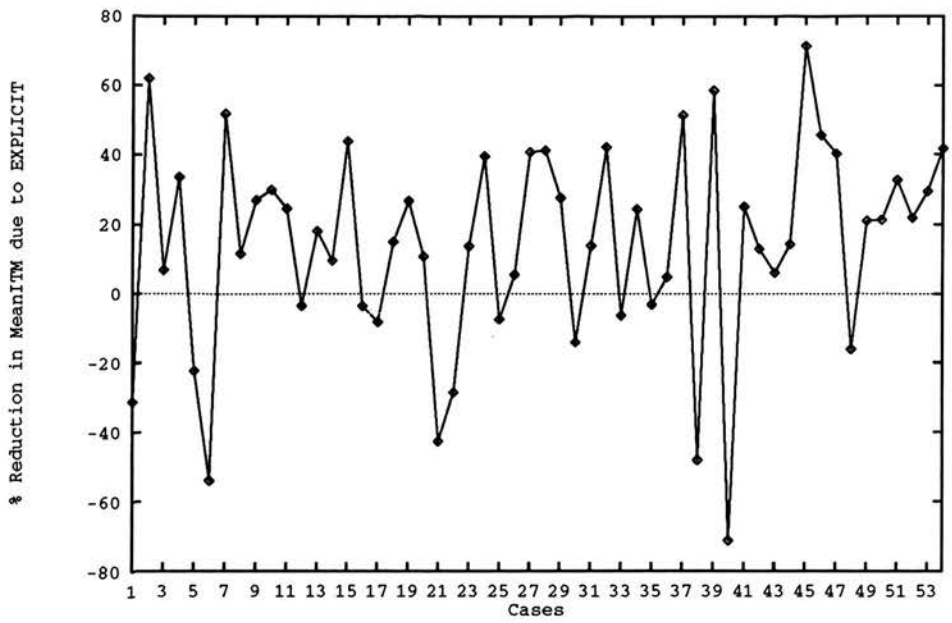

Figure 9.33: Percentage Reduction in MeanITM (EXPLICIT vs. MAXSLK)

9.3.5 Time Performance

The code that implements the versions EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN is not optimised. It includes several output procedures to keep record of all the tasks performed by the different agents and it is implemented in interpreted LISP. Nevertheless, in order to have an idea of the behaviour of the two approaches in terms of times, especially as the size of the problems increase, the time performance of the system was analysed considering the current suboptimised cpu times. An analysis in terms of the number of iterations is also presented.

Analysis in terms of iterations

The objective of this section is to analyse the number of iterations required for each case. Each iteration corresponds to a complete cycle as defined in chapter 5. A comparison between EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN is performed in terms of the number of iterations required to achieve the solution vs. the respective number of jobs, for the different cases.

Figures 9.34, 9.35 and 9.36 display the behaviour of EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN in terms of the number of iterations vs. number of jobs, respectively for type A, type B and type C jobs. Appendix E lists the data that corresponds to the graphs.

From the analysis of the graphs no special correlation between the number of iterations and the number of jobs can be detected, in both cases, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN. Tables 9.14, 9.15 and 9.16 summarise some relevant location and dispersion measures with respect to the number of iterations required to achieve a soltion for each case. The measures in the tables are grouped per type of jobs, A, B and C, and considering the following cases:

- all - for each type, all the cases are considered (cases 1 to 18 *inc.*, for type A jobs; cases 19 to 36 *inc.*, for type B jobs; cases 37 to 54 *inc.*, for type C jobs)
- 2 machines - only the cases that included 2 machines per aggregate resource are considered (cases 1 to 12 *inc.*, for type A jobs; cases 19 to 30 *inc.*, for type B

jobs; cases 37 to 48 *inc.*, for type C jobs)

- 3 machines - only the cases that included 3 machines per aggregate resource are considered (cases 13 to 15 *inc.*, for type A jobs; cases 31 to 33 *inc.*, for type B jobs; cases 49 to 51 *inc.*, for type C jobs)

- 5 machines - only the cases that included 5 machines per aggregate resource are considered (cases 16 to 18 *inc.*, for type A jobs; cases 34 to 36 *inc.*, for type B jobs; cases 52 to 54 *inc.*, for type C jobs)

From the analysis of the graphs and the location and dispersion measures, several inferences can be drawn. The most remarkable aspect is that, considering both versions, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, the *range* and the *maximum* number of iterations required to achieve a solution, for the three different types of jobs and for all the cases, are *very small*: 5 and 11 for EXPLICIT-OAS-OUT and 6 and 12 for EXPLICIT-OAS-IN , for type A jobs; 4 and 9 for the EXPLICIT-OAS-OUT and 10 and 15 for the EXPLICIT-OAS-IN , for type B jobs; 4 and 8 for the EXPLICIT-OAS-OUT and 8 and 13 for the EXPLICIT-OAS-IN , for type C jobs. Recall that the range of jobs in each case goes from 6 to 29 for the three types of jobs. One would expect a greater increase in the number of iterations required to achieve a solution as the number of jobs increases.

The solutions generated by EXPLICIT-OAS-IN seem to display a greater *dispersion* in terms of the number of iterations required to achieve a solution than the solutions generated by EXPLICIT-OAS-OUT, as reflected by the values of the *range* and the *standard deviation.* However, both approaches have a similar behaviour on average, as reflected by the *mean*, 7.556 and 7.389 for type A jobs, respectively for EXPLICIT-OAS-OUT and for EXPLICIT-OAS-IN and considering all the cases, 6.889 and 7.833 for type B jobs, respectively for EXPLICIT-OAS-OUT and for EXPLICIT-OAS-IN and considering all the cases and 6.556 and 7.722 for type C jobs, respectively for EXPLICIT-OAS-OUT and for EXPLICIT-OAS-IN and considering all the cases.

If one analyses the different cases for the number of machines per aggregate resource, the behaviour of both approaches for each subgroup is not very different from the one

exhibited when considering all the cases. The mean of the number of iterations required to achieve the solution oscillates from 6.333 to 7.667 for the case of 2 machines and considering the three types of jobs, from 6.667 to 8.667 for the case of 3 machines and considering the three types of jobs, from 7.0 to 8.333 for the case of 5 machines and considering the three types of jobs. Recall that for the case of 2 machines the number of jobs ranges from 6 to 20, for the case of 3 machines it ranges from 20 to 24 and for the case of 5 machines from 26 to 29. There seems to be an increase of the *mean* of the number of iterations required to achieve the solution as the number of jobs increases, even with the increase of the number of machines, though this increase is small.

The global comparison of EXPLICIT-OAS-OUT with EXPLICIT-OAS-IN in terms of the number of iterations required to achieve a solution is synthesised in table 9.17. This table shows the results for both versions , EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, considering all of the 54 cases. For both versions, the *mean* of the number of iterations required to achieve a solution is very small, 7.0 and 7.648 for EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN respectively. Both versions, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, have a similar behaviour reflected not only on a similar *mean* but also on the *standard deviation* and *median*, when considering the 54 cases.

Cases	*Version*	*Range*	*Minimum*	*Maximum*	*Mean*	*Median*	*SD*
All	OAS-OUT	5	6	11	7.556	7.0	1.723
	OAS-IN	6	6	12	7.389	7.0	1.754
2	OAS-OUT	5	6	11	7.583	6.5	1.929
Machines	OAS-IN	4	6	10	7.167	6.5	1.586
3	OAS-OUT	3	7	10	8.0	7.0	1.732
Machines	OAS-IN	1	7	8	7.333	7.0	0.577
5	OAS-OUT	2	6	8	7.0	7.0	1.00
Machines	OAS-IN	6	6	12	8.333	7.0	3.215

Table 9.14: Location and dispersion measures of the number of iterations, EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT - type A jobs

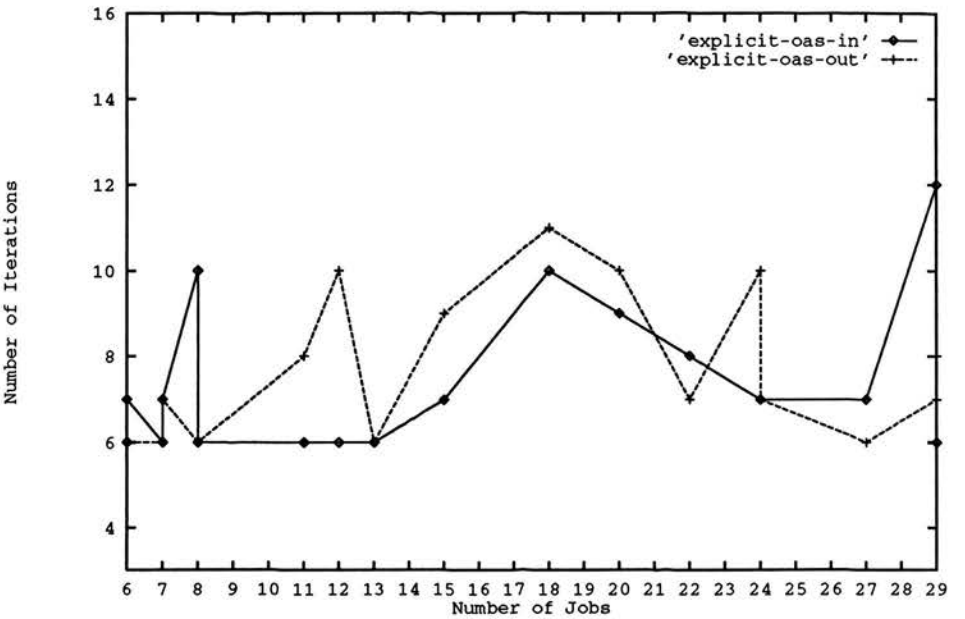

Figure 9.34: Number of Iterations vs. Number of Jobs - type A jobs

Cases	*Version*	*Range*	*Minimum*	*Maximum*	*Mean*	*Median*	*SD*
All	OAS-OUT	4	5	9	6.889	7.0	1.183
	OAS-IN	10	5	15	7.833	7.0	2.383
2	OAS-OUT	4	5	9	6.667	6.0	1.231
Machines	OAS-IN	10	5	15	7.500	6.5	2.812
3	OAS-OUT	3	6	9	7.667	8.0	1.528
Machines	OAS-IN	3	7	10	8.667	9	1.528
5	OAS-OUT	0	7	7	7.0	7.0	0
Machines	OAS-IN	1	8	9	8.333	8.0	0.557

Table 9.15: Location and dispersion measures of the number of iterations, EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT - type B jobs

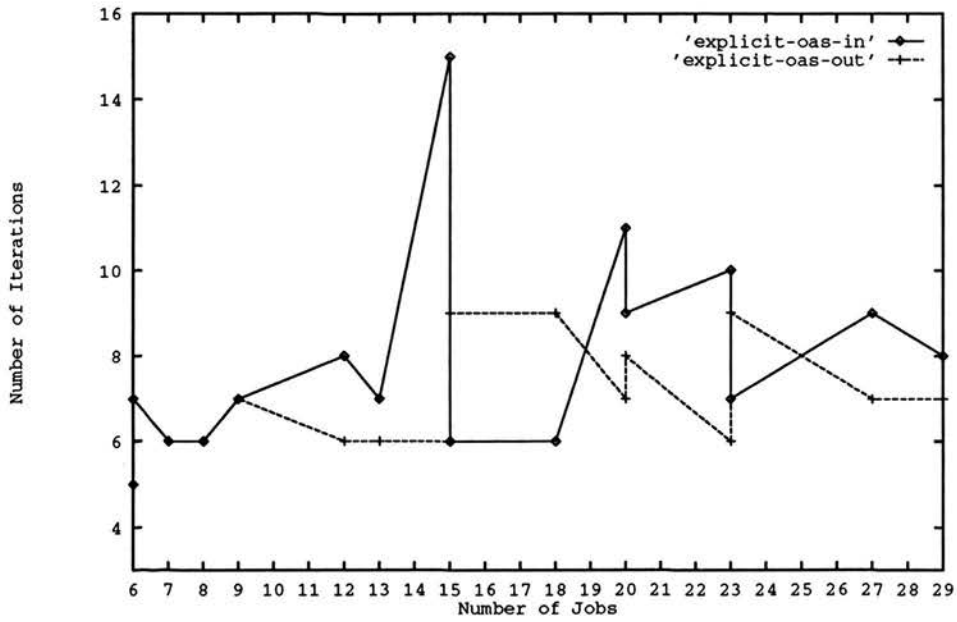

Figure 9.35: Number of Iterations vs. Number of Jobs - type B jobs

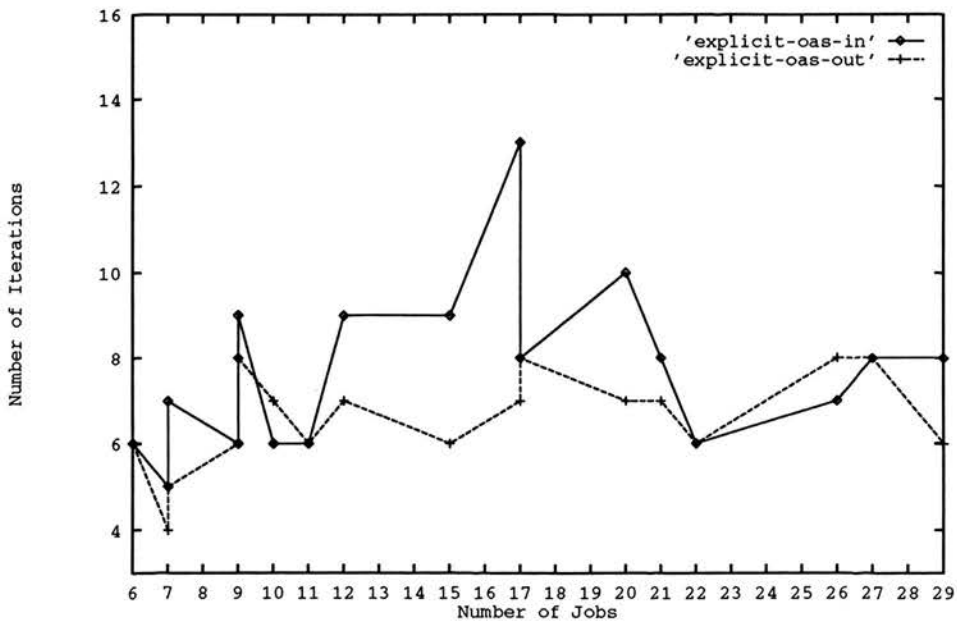

Figure 9.36: Number of Iterations vs. Number of Jobs - type C jobs

Cases	*Version*	*Range*	*Minimum*	*Maximum*	*Mean*	*Median*	*SD*
All	OAS-OUT	4	4	8	6.556	6.5	1.097
	OAS-IN	8	5	13	7.722	8.0	1.904
2	OAS-OUT	4	4	8	6.333	6.0	1.155
Machines	OAS-IN	8	5	13	7.667	7.5	2.188
3	OAS-OUT	1	6	7	6.667	7.0	0.577
Machines	OAS-IN	4	6	10	8.0	8.0	2.000
5	OAS-OUT	2	6	8	7.333	8.0	1.155
Machines	OAS-IN	1	7	8	7.667	8.0	0.577

Table 9.16: Location and dispersion measures of the number of iterations, EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT - type C jobs

Version	*Range*	*Minimum*	*Maximum*	*Mean*	*Median*	*SD*
OAS-OUT	7	4	11	7.000	7	1.401
OAS-IN	10	5	15	7.648	7	2.001

Table 9.17: Location and dispersion measures of the number of iterations, EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT - all types of jobs

CPU time analysis

The current implementation was designed as a proof of concept rather than an attempt at efficiency. Extensive debugging aids, record keeping, including several sorting routines as part of the record keeping process, hamper its efficiency. Furthermore, the entire system was run under an interpreted LISP. Rule of thumb estimates for a compiled version are at least a tenfold increase in execution speed, compared to the interpreted version. Nevertheless, it is worth analysing the time performance of both versions, EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, for the sake of reference and also to compare the behaviour of the two versions.

In this section the time performance of EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT is analysed. Analogously to what was done in terms of the number of iterations required to generate a solution, some descriptive measures of the cpu times are analysed. In order to identify the correlation between the cpu time and the size of the problems, a regression analysis is also presented.

Descriptive Analysis Tables 9.18, 9.19 and 9.20 display descriptive measures in terms of location and dispersion of the cpu time in seconds, with respect to both versions EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT.

Several inferences can be drawn from the analysis of these tables. The *mean* and the *standard deviation* of the cpu time decrease when comparing type A jobs with type B jobs and type B jobs with type C jobs. This is true for the two versions, EXPLICIT-OAS-IN and OAS-OUT, and for the different cases considered in the tables. Considering all the cases, the average cpu time in seconds is 464.082 and 622.213, 354.786 and 477.345 and 210.022 and 357.558, for type A jobs, version EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, type B jobs, version EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, and type C, version EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, respectively. Recall that in type A jobs, the order of the operations is very rigid, while in type C jobs, apart from the first and last operations, the other operations do not have a predefined order. Type B jobs corresponds to a intermediate situation. Because the order of operations is not very

rigid in type C jobs, that means that less conflicts among operations occur.

Considering the different cases shown in each table, there is a large increase of the *mean* of the cpu time when comparing the 2 machines cases with the 3 and 5 machines cases, which is easily justified by the fact that the 3 and 5 machines cases correspond to the cases of more than 20 and 25 jobs per case, respectively. In fact, and in contrast to what was observed to the number of iterations required to achieve a solution, there is a significant discrepancy (see the *range*) between the cpu time required to generate a solution for the 6 jobs cases and the 29 jobs cases, 1627.753 secs and 1698.833 secs, 1433.947 and 1516.583, 1079.787 and 1302.453, for type A jobs, version EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, type B jobs, version EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT and type C jobs, version EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, respectively. Note that, though the cpu time increases as the number of jobs increases, which is reflected on the increase of the *mean* of the cpu time from one class to another, when comparing the relative increase of the *mean* of the cpu time from class to class, one can observe that the increase in the cpu time slows down, which is explained by the increase in the number of machines. Moreover, as with what was observed with the *mean*, the *range* and *standard deviation* of the cpu time decrease when comparing type A jobs with type B jobs and type B jobs with type C jobs.

Table 9.21 compares the performance of EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN considering the 54 cases. On average, EXPLICIT-OAS-IN requires more cpu time to achieve a solution than EXPLICIT-OAS-OUT. Recall that version EXPLICIT-OAS-IN is provided with Operational Agents, which explains the additional cpu time required to achieve a solution. The *mean* of the cpu time required to produce a solution, when applying EXPLICIT-OAS-OUT and considering all the cases is 342.963, while EXPLICIT-OAS-IN requires 485.705 to produce a solution, on average, about 41% more than EXPLICIT-OAS-OUT. This percentage corresponded to 34%, 34% and 70% when considering the type A, type B and type C jobs respectively. The increase in the cpu time from EXPLICIT-OAS-OUT to EXPLICIT-OAS-IN is larger when considering type C jobs. The reason for this phenomenon might be that, though type C jobs are less constrained and so a solution is easier to find, for the same reason more optimisation is required. The

solutions produced by EXPLICIT-OAS-IN exhibit a greater dispersion in terms of the cpu time than the solutions generated by EXPLICIT-OAS-OUT (536.373 vs. 421.877).

Cases	*Version*	*Range*	*Minimum*	*Maximum*	*Mean*	*SD*
All	OAS-OUT	1627.753	10.817	1638.570	464.082	512.999
	OAS-IN	1698.833	17.167	1716.000	622.213	645.404
2	OAS-OUT	714.050	10.817	724.867	157.322	225.099
Machines	OAS-IN	1009.753	17.167	1026.920	223.239	319.6900
3	OAS-OUT	352.667	705.283	1057.950	895.883	178.057
Machines	OAS-IN	262.180	1136.700	1398.880	1262.103	131.460
5	OAS-OUT	729.437	909.133	1638.570	1259.318	365.586
Machines	OAS-IN	289.270	1426.730	1716.000	1578.220	145.122

Table 9.18: Location and dispersion measures of the CPU time (seconds), EXPLICIT-OAS-OUT vs. EXPLICIT-OAS-IN - type A jobs

Cases	*Version*	*Range*	*Minimum*	*Maximum*	*Mean*	*SD*
All	OAS-OUT	1433.947	9.5331	1443.480	354.786	420.030
	OAS-IN	1516.583	17.217	1533.800	477.345	532.410
2	OAS-OUT	361.467	9.533	371.000	115.021	133.060
Machines	OAS-IN	453.283	17.217	470.500	155.844	159.951
3	OAS-OUT	665.417	262.800	928.217	657.022	349.352
Machines	OAS-IN	882.750	371.950	1254.700	936.040	489.887
5	OAS-OUT	753.813	689.667	1443.480	1011.610	388.743
Machines	OAS-IN	507.750	1026.050	1533.800	1304.650	257.462

Table 9.19: Location and dispersion measures of CPU time (seconds), EXPLICIT-OAS-IN vs. EXPLICIT-OAS-OUT - type B jobs

Cases	*Version*	*Range*	*Minimum*	*Maximum*	*Mean*	*SD*
All	OAS-OUT	1079.787	8.383	1088.170	210.022	284.417
	OAS-IN	1302.453	15.367	1317.820	357.558	399.669
2	OAS-OUT	247.717	8.383	256.100	61.699	76.552
Machines	OAS-IN	476.350	15.367	491.717	127.354	151.027
3	OAS-OUT	474.883	156.600	631.483	395.233	237.450
Machines	OAS-IN	592.717	376.900	969.617	756.367	329.468
5	OAS-OUT	707.120	381.050	1088.170	618.107	407.092
Machines	OAS-IN	777.503	540.317	1317.820	879.562	398.096

Table 9.20: Location and dispersion measures of the CPU time (seconds), EXPLICIT-OAS-OUT vs. EXPLICIT-OAS-IN - type C jobs

Version	*Range*	*Minimum*	*Maximum*	*Mean*	*SD*
OAS-OUT	1630.187	8.383	1638.570	342.963	421.877
OAS-IN	1700.633	15.367	1716.000	485.705	536.373

Table 9.21: Location and dispersion measures for the CPU time (seconds), EXPLICIT-OAS-OUT vs. EXPLICIT-OAS-IN - all types of jobs

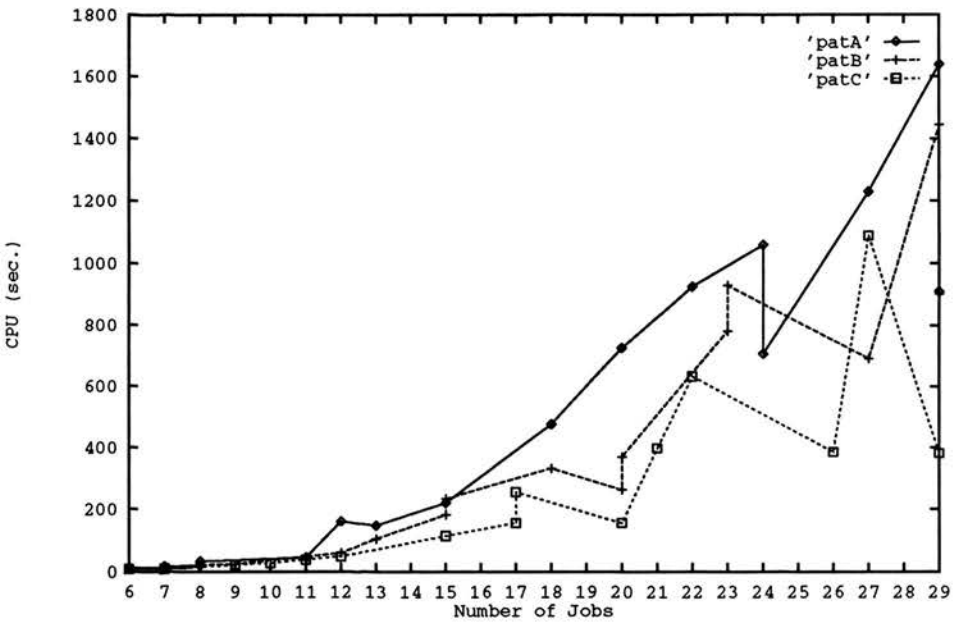

Figure 9.37: CPU time (seconds) vs. Number of Jobs, EXPLICIT-OAS-OUT

Regression Analysis Figure 9.37 and figure 9.38 show the set of all the observations on the number of jobs and the cpu time in seconds with respect to version EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, respectively. From the analysis of these diagrams one can observe:

- the cpu time increases as the number of jobs increases;
- the relation between the cpu time and the number of jobs differs same depending on the type of jobs considered;
- the increase of the number of machines affects the behaviour of the curves, though it is not clear in what way;

In order to investigate the behaviour of versions EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN as the size of the problems increase, a regression analysis was performed. Appendix F describes the results of the regression analysis.

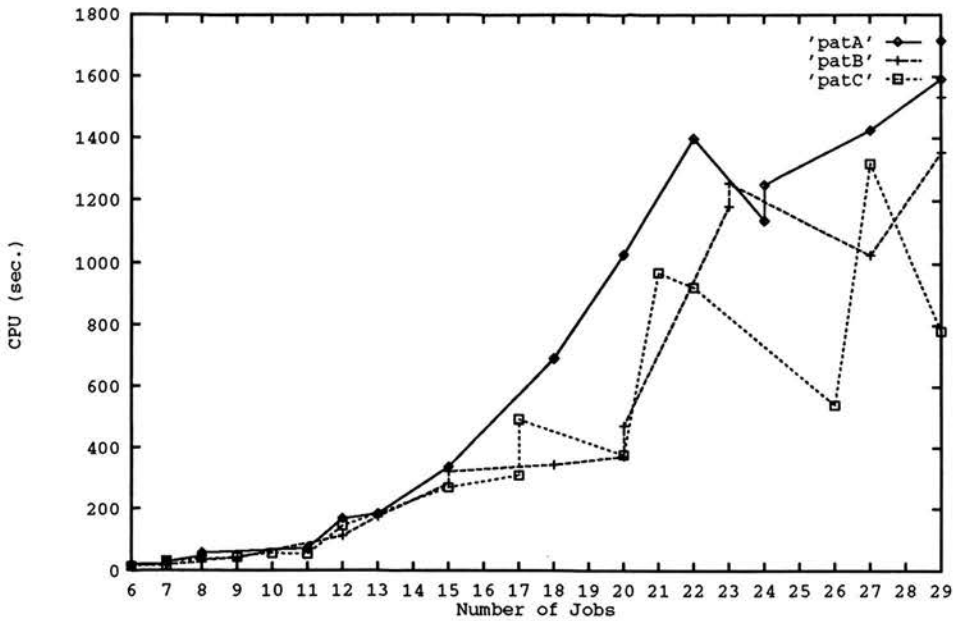

Figure 9.38: CPU time (seconds) vs. Number of Jobs, EXPLICIT-OAS-IN

The the results of the regression analysis are not very conclusive though they seem to indicate that the correlation between the cpu time and the number of jobs follows the power law model, for both versions of EXPLICIT.

As a conclusion, one can consider the results of the cpu time analysis very encouraging - the maximum value of the cpu time is 1716 secs for a 29 jobs problem and the correlation between the cpu time and the number of jobs seems to follow a power law. As mentioned before, the current implementation was designed as a proof of concept rather than an attempt at efficiency. It is possible to improve the efficiency of the current implementation substantially.

9.4 Summary

In this chapter two versions of EXPLICIT, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, were comprehensively described. In order to test the two versions, a battery of 54 cases was generated. The description of the generator of cases was presented in this chapter.

The performance of EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN is analysed considering a set of measures defined in this chapter. EXPLICIT-OAS-IN clearly outperforms EXPLICIT-OAS-OUT with respect to the performance measures adopted for evaluating tardiness and utilisation of resources. In terms of completion times, EXPLICIT-OAS-IN shows a smaller improvement over EXPLICIT-OAS-OUT. Due to its superiority, EXPLICIT-OAS-IN is tested against four dispatch rules: SOF, MOF, MINSLK and MAXSLK.

EXPLICIT clearly outperforms the four dispatch rules in terms of tardiness. MINSLK is the dispatch rule that displays a behaviour closer to the performance of EXPLICIT in terms of tardiness. With respect to completion times, the dispatch rule MOF performs similarly to EXPLICIT-OAS-IN, though EXPLICIT slightly outperforms MOF. Regarding the utilisation of resources, EXPLICIT performs better than MINSLK, MAXSLK and SOF. The heuristic MOF performs better than EXPLICIT. However, MOF performs very poorly in terms of tardiness. As a conclusion, considering all the measures, EXPLICIT-OAS-IN outperforms the dispatch rules, though the superiority of EXPLICIT over the selected dispatch rules is more marked in terms of tardiness. These results are not a surprise since the utility function assigned to the Resource Tactical Agents considers as first objective the meeting of due-dates and as a secondary objective the utilisation of the resources and EXPLICIT (or more rigorously EXPLICIT-OAS-IN) is provided with Operational Agents whose expertise is to minimise the maximum lateness of the jobs assigned to their machines and, implicitly, to guarantee a better utilisation of the machines.

Regarding the number of iterations required to generate a solution, it is remarkable that, for both version, the *mean* and *range* of the number of iterations required to achieve a solution is very small, 7 and 7 for EXPLICIT-OAS-OUT and 7.648 and 10 for EXPLICIT-OAS-IN. Both versions, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, have a similar behaviour reflected not only on a similar *mean* but also on the *standard deviation* and *median.*

In terms of the cpu time analysis, very encouraging results were obtained as well, not only in terms of the regression analysis but also considering the small magnitude of the cpu time required to produce a solution when applying both versions EXPLICIT-OAS-

OUT and EXPLICIT-OAS-IN. According to the regression analysis results, the power law model seems to be the best fit for the generated sample (54 cases) and the maximum value of the cpu time required to produce a solution was 1716 secs for a 29 jobs problem. As mentioned above, the current implementation was not designed as a proof of efficiency and it is possible to improve its efficiency substantially.

Chapter 10

Conclusions and Future Work

This chapter summarises the main contribution of the research reported in this thesis. A discussion of possibilities for future research is also included.

10.1 An Approach to Job Shop Scheduling Exploiting Conflict

10.1.1 Concepts

The Organisation Metaphor

EXPLICIT is conceived as an *Organisation.* This concept is not new in itself. The structure adopted for EXPLICIT was inspired by other existing systems, particularly DAS [Buchanan *et al* 88], [Buchanan *et al* 89], [Burke & Prosser 89], [Burke 89]. The contribution embodied in EXPLICIT is to push the *Organisation* metaphor one step further to cope with the complexity of the job scheduling problem. In EXPLICIT the functions of *analysis, planning and evaluation* are reinforced and explicitly assigned to the different agents of the system. Agents are assigned different roles and functions depending on their position within the structure of the *Organisation.* Top level agents are responsible for global and structural functions (and consequently decisions), while more detailed and specific tasks and functions are assigned to the bottom level agents. In this *Organisation*, agents at the same level state their interests independently of each other. The strategy adopted is "divide and conquer".

Conflict

The idea of *Conflict* is also not new (e.g., DAS). However, in EXPLICIT, it is recognised that *conflict* as a consequence of the scheduling process should be exploited as a way of guaranteeing *pluralism* and of guaranteeing that agents make their best scheduling decisions without influencing each other. In EXPLICIT, the different Tactical Agents perform their scheduling decisions without being aware of the decisions performed by other Tactical Agents. In particular, Resource Tactical Agents assign times and individual machines to operations (with or without the intervention of the operational Agents) without any form of communication between each other. The only information that the Resource Tactical Agents have about each operation is its time windows. Whenever Resource Tactical Agents, with or without the Operational Agents, assign times and machines to operations, there is no mechanism to propagate that effect immediately throughout the other operations of the same job. The idea embodied in EXPLICIT is to allow Tactical Agents (Job Tactical Agents and Resource Tactical Agents) to formulate their best scheduling decisions considering their own interests. On the other hand, the Strategic Agent is provided with mechanisms to analyse and evaluate the criticality of conflicts, the criticality of each operation per se and, based on that, the Strategic Agent defines plans for conflict resolution. EXPLICIT embodies a strategy for *achieving global coherence by exploiting conflict.*

Pluralism vs. Integration of Multiple Scheduling Perspectives

[Kornfeld & Hewitt 88] introduced the idea of the "scientific community metaphor". The success of scientific research depends heavily on complementary perspectives and knowledge, in order to allow *pluralism.* In EXPLICIT ensuring *pluralism* corresponds to integrating different scheduling perspectives. *Pluralism* is achieved by allowing agents to perform their scheduling decisions independently of each other, by providing agents with both empirical knowledge (heuristics, dispatch rules) and theoretical knowledge (optimal algorithms) and by explicitly allowing the coexistence of scheduling perspectives. ISIS and SOJA use essentially a *job based perspective*, while RESS-II, OPT and DAS

are mainly *resource based* schedulers. OPIS and SONIA interleave a *job based perspective* with a *resource based perspective.* This strategy of scheduling has been named *opportunistic scheduling.* EXPLICIT also interleaves a *job based perspective* with a *resource based perspective.* However, EXPLICIT goes one step further in the sense that, though the *job based perspective* and the *resource based perspective* are combined to analyse the complexity of the scheduling problems, decisions are made considering as focus of attention the criticality of each operation per se (*operation based perspective*). In other words, Job Tactical Agents perform their scheduling decisons considering a *job based perspective* while Resource Tactical Agenst adopt a *resource based perspective.* However, since the Tactical Agents perform their scheduling decisions asynchronously, conflicts are likely to occur. The Strategic Agent analyses the conflicts and it performs scheduling decisions based on the criticality of each operation. The Strategic Agent enforces an *operation based perspective*, which results from the combination of the *job based perspective* and the *resource based perspective.* To an extent this approach has some similarities with the approaches adopted by [Sadeh 91] in MICRO-BOSS and by [Berry 91].

Structural Awareness

In a distributed and conflicting environment it is essential to provide the different agents of the system with a set of mechanisms to make them aware of the structural and intrinsic properties of the (sub)problems that they have to solve and the interaction of their (sub)problems, without relying on communication with each other. The underlying rationale for providing such mechanisms to a system is that the inherent structural properties of a domain can be used to alleviate its "intractability", in the sense that the whole search space is decomposed into smaller search spaces. In EXPLICIT several mechanisms are provided to the agents to ensure *structural awareness.*

Merge of OR and AI Techniques - Increasing Optimisation

EXPLICIT integrates optimal models to solve the mathematically well defined components of the (sub)problems its agents have to solve with more heuristic type procedures for the more qualitative components of the system. EXPLICIT combines OR and AI techniques in a useful way in order to enhance *optimisation* rather than settling for *satisfaction.*

10.1.2 Algorithms

A Distributed Approach to Job Shop Scheduling Exploiting *Conflict*

The more tangible contribution of the research reported in this thesis is a set of concepts and algorithms, which include:

- *Conflict* vs. *Conflict Resolution Mechanism* - a set of concepts and algorithm to deal with *conflict*:
  - Formalisation of the concept of *conflict* - the concept of *conflict* is formally defined. Different type of *conflicts* and their properties are defined. Rules of dependency between *conflicts* are defined as well as rules of dominance of *conflicts.*
  - Algorithms for *conflict detection* and *conflict resolution* combining the concept of *conflict*, its types and properties.
- An algorithm for the assignment of times and machines to operations
  - An assignment based algorithm for resource allocation for the job shop scheduling problem - this algorithm consists of a sequence of assignment problems.
  - A utility function - a particular utility function for the set of assignment problems (APs) performed for the assignment of times and machines to operations.

- An approach which integrates a *job based scheduling* perspective with a *resource based scheduling* perspective resulting in an *operation based scheduling* perspective.

A Distributed Problem Solving (DPS) approach is adopted in EXPLICIT. EXPLICIT can be compared to a hierarchical organisation with three main levels: the Strategic level, the Tactical level and the Operational level. This structure is inspired by DAS ([Burke 89], [Buchanan *et al* 89], [Burke & Prosser 89]). However, although there are some similarities between EXPLICIT and DAS in terms of the general structure of the system, there are substantial differences in terms of the processes associated with the different agents of the systems, i.e., the functional organisation of the systems, and in terms of the techniques and the methods used in both systems.

The approach adopted in EXPLICIT aims to integrate multiple scheduling perspectives, to integrate predictive capabilities with reactive capabilities and to increase optimisation rather than settling for satisfaction. The underlying principles for tackling these goals are: the assignment of different functional roles and skills to agents within a distributed problem solver framework; ensuring *pluralism*; the merge of OR and AI techniques; ensuring *contextual awareness*; providing agents with mechanisms to detect and solve *Conflict*. Considering the DPS point of view, the research reported in this thesis investigates strategies for ensuring the *Global Coherence* and *Conflict Resolution* within a distributed problem solving environment. The approach adopted in EXPLICIT recognises that *Conflict* as a consequence of the scheduling process should be exploited as a way of allowing the different agents of the system to freely express their interests. *The thesis is that not only is it important to deal with Conflict, Conflict should be exploited to increase Global Coherence.*

The scheduling process adopted in EXPLICIT consists of: analysis of the problem and detection of conflicts performed by the Strategic Agent; assignment of time windows performed by the Job Tactical Agents; assignment of start times and individual machines performed by the Resource Tactical Agents. Operational Agents are responsible for optimising the schedules suggested by the Resource Tactical Agents. The concept of conflict is a central concept in EXPLICIT. The idea embodied in EXPLICIT is to allow

Tactical Agents (Job Tactical Agents and Resource Tactical Agents) to formulate their best scheduling decisions considering their own interests and independently of each other. The Strategic Agent, on the other hand, is provided with mechanisms to analyse and evaluate the criticality of conflicts, the criticality of each operation per se and, based on that, the Strategic Agent defines plans for conflict resolution. EXPLICIT combines a conflict resolution algorithm, a critical path method, CPM, (see, e.g., [Willis 85], [Davis & Patterson 75] and [Fendley 68]), an assignment based algorithm, which integrates an assignment algorithm (see, e.g., [Gondran & Minoux 84]) and an optimal algorithm for minimising maximum lateness ([McMahon & Florian 75]). EXPLICIT performs a sequence of optimisations of the load balancing of the resources, each time assigning times and resources to the operations considered more critical. Each operation is analysed individually (*operation based perspective*) by the Strategic Agent and its criticality results from the combination of a *job based perspective*, which is put into action by the Job Tactical Agents, with a *resource based perspective*, which is put into action by the Resource Tactical Agents.

The results of the comparison of EXPLICIT with four popular dispatch rules are very encouraging. Considering all the selected performance measures, EXPLICIT outperforms the dispatch rules, particularly in terms of tardiness and utilisation of resources. These results are not surprising since the utility function assigned to the Resource Tactical Agents considers as its first objective the meeting of due-dates and, as its secondary objective, the utilisation of the resources. Furthermore, EXPLICIT (or more precisely EXPLICIT-OAS-IN[1]) is provided with Operational Agents whose expertise is to minimise the maximum lateness of the jobs assigned to their machines and, implicitly, to guarantee better utilisation of the machines.

Regarding the number of iterations required to generate a solution, it is remarkable that, for both versions, the *mean* and the *range* of the number of iterations required to achieve a solution is very small. In terms of the cpu time analysis, very encouraging results were obtained as well, considering the relative small magnitude of the cpu time required to produce a solution when applying both versions EXPLICIT-OAS-OUT and

[1]EXPLICIT-OAS-IN is the version of EXPLICIT *with* Operational Agents and EXPLICIT-OAS-OUT is the version of EXPLICIT *without* Operational Agents.

EXPLICIT-OAS-IN. The current implementation of EXPLICIT was designed as a proof of concept rather than as an attempt at efficiency. It is possible to improve its efficiency substantially.

10.2 Future Research

There are a number of ways in which the research reported in this thesis can be extended, some of which are briefly described below.

10.2.1 A Physically Distributed System

EXPLICIT is conceptually distributed but implemented on a sequential machine. A natural extension to EXPLICIT is to implement it in a physically distributed environment. The results obtained in terms of the number of iterations and the cpu time required to achieve a solution are very encouraging, in particular the small magnitude of the average number of iterations required to generate the final schedule. It provides an indication that the performance of EXPLICIT could be improved if a physically distributed environment was adopted.

In the current implementation of EXPLICIT, whenever the Strategic Agent replans the work to be assigned to the Resource Tactical Agents, each Resource Tactical Agent involved in the plan performs its rescheduling tasks one at a time. In a physically distributed environment, all the Resource Tactical Agents involved in the plan can execute their rescheduling tasks concurrently. Furthermore , Job Tactical Agents and Resource Tactical Agents and Operational Tactical Agents can perform their scheduling operations concurrently.

The scheduling results which would be obtained in a physically distributed version of EXPLICIT would be identical to the results obtained with the current implementation of EXPLICIT, if the same mechanisms actually implemented in terms of dependencies and dominance of conflicts were used. The current version of EXPLICIT includes mechanisms to keep track of dependencies between sub-problems, and so, different sub-problems can be performed concurrently. The concept of *conflict* and particularly the rules of

dependencies between *conflicts* and dominance of *conflicts* reflected in the attributes *parentgenerator*, *conflictgenerator* and *parents* constitute mechanisms to keep track of the decisions performed by the Strategic Agent in order to solve the *conflicts*. If some of the decisions taken by the Strategic Agent are revealed to be poor decisions later, the attributes *parentgenerator*, *conflictgenerator* and *parents* provide the necessary information for the Strategic Agent to undo those decisions.

In a physically distributed version of EXPLICIT time performance is expected to improve. As mentioned above, the results obtained in terms of the number of iterations, in particular the small magnitude of the average number of iterations required to achieve a solution, provide an indication that the performance of EXPLICIT can be improved if a physically distributed environment is adopted.

Other forms of infusing parallelism in EXPLICIT are worth investigating if implementing EXPLICIT in a parallel machine particularly at the level of the algorithms. Other forms of conflict resolution can also be explored (see below for an outline of a least commitment strategy for conflict resolution).

10.2.2 Problem Solving Agents

Least Commitment Strategies for Conflict Resolution

The current rules of dominance of *conflicts* embodied in EXPLICIT state that if a *conflict generator* and a *level reaction conflict* involve the same operation, the *level reaction conflict* has to be cancelled, since the Strategic Agent cannot have in its final plan two *conflicts* involving the same operation. As mentioned in chapter 5, the opposite rule could be adopted, which would imply a least commitment strategy. The idea is that if a set of operations to be performed on a certain aggregate resource (i.e., *level reaction conflicts*) have to be rescheduled due to a certain conflict (the *chain reaction conflict* that is the parent of the *level reaction conflicts*) , existing *conflict generators* involving operations on that resource that have to be rescheduled anyway should be cancelled because the new rescheduling might mean a new opportunity for the operations involved in the *conflict generators*. In other words, the original time windows

of the *conflict generators* might be able to stay unchanged, due to the change of the time windows of other operations on the same resource (the ones involved in the *level reaction conflicts*). So, whenever a *level reaction conflict* and a *conflict generator* exist on a certain operation, if the *level reaction conflict* dominates the *conflict generator* the operation involved in the conflict is rescheduled (together with other *level reaction conflicts*) with the same previous time windows (instead of having them changed as in the case of *conflict generator*). Due to the rescheduling of other operations on the same resource the circumstances that had made that operation a *conflict generator* might have changed. This approach involves a later commitment strategy, which means a more opportunistic approach. The other side of the coin if this strategy is adopted is that the generation of a schedule might require more computational resources. Nevertheless, it seems worth exploring this strategy. Actually, some experimental tests using this strategy seem to indicate an improvement of the quality of the solution, though the time required to generate the solution slightly increased.

The exploration of other strategies for *conflict resolution* constitutes another area for further research.

Different Utility Functions Assigned to the Tactical Agents

A particular utility function was defined for the assignment problems performed by the Resource Tactical Agents as part of the assignment based algorithm. As mentioned in chapter 5, depending on the particular problem, on the particular domain and on the particular objectives considered, different utility functions can be conceived. The Job Shop Scheduling problem is characterised by several goals and constraints, usually interacting in conflicting and unforeseen ways, most of them with a qualitative and subjective nature. The identification of the different scheduling objectives and how to reflect them in the utility function of each assignment problem constitute an area for further research.

Related to the definition of alternative utility functions is the investigation of new algorithms for the assignment of times and machines to the operations. For instance,

an obvious idea is that some efficient algorithms reported in the literature might be suitable for a particular domain or problem in those cases where the Resource Tactical Agent is responsible for one, two or three machines (see e.g., [Kan 76], [French 82], [Blazewicz88 *et al* 88], [Buxey 89], [Cheng & Sin 90], [Blazwicz *et al* 91]).

Different Optimisation Functions Assigned to the Operational Agents

The algorithm that was assigned to the Operational Agents was an optimal algorithm for minimising maximum lateness. Although the problem of sequencing a set of jobs in a single machine is NP-complete in the strong sense [Garey & Johnson 79], the algorithm that minimises the maximum lateness of a set of jobs on a single machine ([McMahon & Florian 75]) has been reported as being capable of solving fairly large problems in a matter of seconds with data drawn from a realistic range. [Kan 76] reports favourable results with the algorithm of [McMahon & Florian 75] on problems with up to 80 jobs.

Another algorithm similar to the algorithm of [McMahon & Florian 75] was devised by [Carlier 82]. [Carlier 82] achieved good results with his algorithm on problems with up to 1000 jobs. This algorithm differs from the algorithm of [McMahon & Florian 75] in the way the next node of the branch and bound tree is selected. At every node of the branch and bound tree, a heuristic based on the MWKR (most work remaining) priority dispatching rule is applied to the current one-machine problem.

Other optimisation functions can be assigned to the Operational Agents depending on the particular problem, on the particular domain and on the particular objectives considered. Other algorithms can be provided to the Operational Agent considering other objectives and considering the current state of the art of the one-machine scheduling algorithms (see, e.g., [Kan 76], [French 82], [Gupta & Kyparisis 87], [Dileepan & Sen 88], [Blazwicz *et al* 91]).

10.2.3 A More Robust and More Efficient Implementation

Earlier it was mentioned that the current implementation of EXPLICIT was designed as a proof of concept rather than an as attempt at efficiency. Extensive debugging aids, record keeping, including several sorting routines as part of the record keeping process, hamper its efficiency. Furthermore, the entire system was run under an interpreted LISP. Rule of thumb estimates for a compiled version are at least a tenfold increase in execution speed, compared to the interpreted version. A basic implementational task would be to make EXPLICIT more robust and more efficient.

10.2.4 Expansion of EXPLICIT to Cope with Other Problems

In Job Shop Scheduling

Real world job shop scheduling problems are very complex. In particular, there are numerous factors that influence the generation of schedules - set-ups, preemption, preferences, break downs, preventive maintenance, machines with capacity greater than one, among many others factors, and this list does not consider the aspects related to the stochastic factors that influence either the demand for jobs or the supply in terms of resources (e.g., break downs).

EXPLICIT was conceived to tackle a variety of job shop scheduling problems. The idea was to develop a framework flexible enough to cover a large spectrum of problems. For particular problems with specific characteristics, the system can be provided with more accurate processes. EXPLICIT has a very modular, flexible and expandable representation that allows the coexistence of qualitative reasoning type modules with more mathematical reasoning type components. Expanding EXPLICIT to cope with some of the aspects mentioned above constitutes another area of research.

Other Scheduling Problems

A more ambitious idea in terms of future work is the expansion of the framework embodied in EXPLICIT to cope with other types of scheduling problems, e.g., trans-

portation scheduling and timetabling.

Appendix A

Sequencing Jobs on A Single Machine

This appendix describes an algorithm for sequencing jobs on a single machine, minimising maximum lateness, due to [McMahon & Florian 75]. This algorithm is assigned to the Operational Agents in the version EXPLICIT-OAS-IN (see chapter 5 and chapter 9).

Optimal Algorithm for Sequencing Jobs on A Single Machine

The algorithm developed by [McMahon & Florian 75] is an implicit enumeration algorithm that uses a branch-and-bound technique. It defines a search tree that has the initial solution generated by the Schrage's heuristic [Schrage 71] as the root node. A lower bound is calculated at this node. If the value of the solution is greater than the lower bound that was found, new nodes corresponding to new problems are branched from the root node. For each new node, a new problem, the Schrage's heuristic is applied again, and lower bounds are calculated. If the optimal solution is not discovered among the new problems that resulted from the branching of the root node, new successor nodes are created in a similar manner for each node. This algorithm may be called a "primal" branch-and-bound method, since a complete solution is associated with each node of the search tree. The strategy adopted chooses the node with the least lower bound for branching. The computations may terminate at any level, including the root.

Before detailing the steps of the algorithm, let us define how to generate a solution for a problem by applying the Schrage's heuristic, how to compute lower bounds for each problem and how to generate the successor nodes (new problems) of a node to improve the solution.

• Schrage's Heuristic ([Schrage 71])

Let:

r - available time of the machine
a_i - available time of the job i to be processed on the machine

d_i - processing time of the job i on the machine
b_i - due date of the job i
S - set of jobs to be scheduled on the single machine

The steps for generating a solution for the single machine problem according to the Schrage's heuristic are:

1 Set $t \leftarrow r$

2 Is there at least one job $i \in S$ such that $a_i \leq t$? If so goto 4.

3 Set $t \leftarrow min_{i \in S}\ a_i$

4 Among all jobs $i \in S$ such that $a_i \leq t$ choose the job j that has the smallest due date b_j; break ties on due date by selecting the job with the largest duration d_j.

5 Schedule the chosen job next. Set $t \leftarrow (t + d_j)$. Set $S \leftarrow S-$ job j.

6 If $S \neq \emptyset$, go to 2. Otherwise the schedule is complete.

The schedule generated by this heuristic consists of a number of *blocks*. A *block* is a period of continuous utilisation of the machine, such that the last job in the block completes its processing time at a time t, when no other job is delayed. The next job in the sequence will then start a new block. It commences the new block at its ready time, which may be at time t.

• Lower Bounds for a solution generated by the Schrage's heuristic

Lower bounds can be computed while generating the Schrage's heuristic solution.

Let P_j be the set of jobs that precede job j in its block and the job j itself. The completion time of job j, t_j, is the earliest possible completion time of the set jobs P_j, i.e if another order was chosen for the jobs in P_j, the last job in this new order could not complete its processing time before t_j. Let us consider such alternative orders. If job $k \in P_j$ is scheduled last, rather than job j, then two possibilities arise:

- $b_k \leq b_j$ - in such a schedule, if job k is scheduled last, job k would be at least as late as the lateness of job j;
- $b_k > b_j$ - in such a schedule, if job k is scheduled last, job k could not complete its processing before $t_j + 1$, since the order change would leave a gap in the scheduled jobs. The lateness of job k would then be at least $t_j + 1 - b_k$.

Thus, lower bounds for a solution generated by the Schrage's heuristic can be computed as follows:

$$LB_j^1 = \begin{cases} t_j - b_j & \text{if } b_j = max_{i \in P_j}\ b_i \\ t_j + 1 - b_k & \text{if } b_j \neq b_k = max_{i \in P_j}\ b_i \end{cases}$$

$$LB^2 = max_i[a_i + d_i - b_i]$$

Hence,

$$LB = max[max_i \ LB_i^1, LB^2]$$

• Improving a solution generated by the Schrage's heuristic

Let the *critical job* be the job that realises the value of the max lateness, in a given schedule. Any nonoptimal schedule may be improved only if the *critical job* may be rescheduled earlier. Let us consider the block in the solution where the *critical job*, say j, is found. If $b_j = max_{i \in P_j} \ b_i$, then the schedule may not be improved and the initial solution is optimal. If, however, $b_j < b_k$ for $k \in P_j$, then it may be possible to improve the solution by scheduling job k after job j.

Let G_j, *the generating set of job* j, consist of jobs $i \in P_j$, such that their due dates are greater than the due date of the *critical job* j. These are the set of jobs that, if scheduled later than j, may reduce the maximum lateness of the solution. The way to try to improve the solution is to create new problems, one for each job $k \in G_j$. The way to guarantee that each job $k \in G_j$ is scheduled later than job j is to make $a'_k = a_j$. If the Schrage's heuristic is applied to the new problem where $a'_k = a_j$, job k will be scheduled later than job j.

The detailed steps of the implicit enumeration algorithm [McMahon & Florian 75] are as follows. In the search tree for the algorithm, nodes correspond to solutions generated by the Schrage's heuristic for the different problems, starting with the initial problem for the root node and redefining this problem according to the generating sets. In this search tree an *open* node is a node that does not have successors, but may be selected as the next node to be branched from. A *closed* node is a node that cannot have successors.

The Implicit Enumeration Algorithm

• Step 0 - Compute the initial solution applying the Schrage's heuristic;

  • Compute its lower bound, LB

  • Find the *critical job* j and its lateness SOL.

    – if $SOL = LB$ then STOP ; the optimal solution was found

    – if $SOL \neq LB$ then let the node corresponding to the initial solution be the *parent node.*

• Step 1- Find the *generating set*, G_j and create $|G_j|$ new nodes. For each of these nodes, let us say the node that corresponds to job k ($k \in G_j$):

  • Set $a'_k = a_j$ for $k \in G_j$.

  • Recompute the Schrage's heuristic solution for the new problem

- Recompute lower bounds , LB^{loc}
- Find the *critical job* and its lateness SOL^{loc}

• Step 2 - If all the new nodes created in Step 1 have been examined, close the *parent node* and go to Step 5. Otherwise select the node with the smallest SOL^{loc}

• Step 3 - Set $SOL \leftarrow min[SOL, SOL^{loc}]$

- if SOL is less than or equal to the least lower bound of all the open nodes then STOP; the optimal solution was found
- otherwise go to step 4

• Step 4

- if SOL^{loc} equal LB^{loc} then close this node and go to Step 2 ; the solution associated with this node is locally optimal.
- otherwise go to Step 5

• Step 5 Select among the open nodes, the node with the least lower bound; call this node the *parent node*, and go to Step 1.

Appendix B

Conflict Propagation and Plans

This appendix describes the results of conflict propagation and plans related to the newspaper example described in the chapter 7. It includes comprehensive lists in terms of:

- Conflict propagation performed by JTAs - the results of conflict propagation obtained in iteration 2 are included in this appendix[1]. To avoid repetition, the results of conflict propagation obtained in iteration 3 are not included. In the remaining iterations there is no need for conflict propagation since there are no new conflict generators.
- SA's plans for conflict resolution - to illustrate the way the SA elaborates its final plan for conflict resolution, the several intermediate stages of the SA's final plan are listed in this appendix, starting with iteration 2[2]. The "plan-with-lrs" is the plan after the generation of the level reaction conflicts. At this stage all the conflicts among conflicts still exist. The "plan-with-lrs-reds" is the plan with all the conflicts among conflicts removed, except redundant level reaction conflicts. The "final-plan" is the final plan for conflict resolution, after all the conflicts among conflicts have been removed. In this final plan, no two different conflicts can be active on the same operation. The final plan is ordered by increasing proptime. For the remaining iterations, and again to avoid repetion, only the final SA's plan is included.

Let us adopt the following keys:

```
num          - the reference number of the conflict
job          - the job name
```

[1]Note that in iteration 1 the JTAs perform a "fresh" assignment of time windows to operations. The result of the time windows assignment is presented in tables in chapter 7.

[2]Again, in iteration 1 the plan for conflict resolution corresponds to all the operations of all the jobs. The SA plan consists of sending all the operations to the respective JTAs for time windows assignment.

```
res           - the aggregate resource name
job-assign-st - the start time assigned to the job by the corresponding
                RTA
job-in-conf   - is the job involved in a conflict?
num-conf      - the reference number of the conflicts affecting the job
                involved in this conflict
proptime      - the time proposed for the operation by the Strategic
                Agent
propslack     - the slack proposed for the operation by the Strategic
                Agent
newtime       - the new time proposed for the operation
ftime         - the new finish time proposed for the operation
newslack      - the new slack proposed for the operation
type          - the type of the conflict: g - conflict generator; cr -
                chain reaction conflict; lr - level reaction conflict
status        - the status of the conflict: new; sent; cancelled;
                solved
parent-gen    - the reference number of the root conflict that
                originated this conflict
conf-gen      - the reference number of the immediate conflict
                generator that originated this conflict
parent        - the reference numbers of the immediate parents
                conflicts of this conflict (a conflict might have
                several parents)
children      - the reference numbers of the conflict children of this
                conflict (no-children is used to indicate 0 children)
cur-it        - the iteration on which the conflict was detected
```

B.1 Iteration 2

Conflict Propagation

JTA Carla

```
job-propagation-before-conflict-resolution

num 5 job carla res fin job-assign-st 570 job-in-conf yes num-conf 5
proptime 545 propslack 75 newtime 570 ftime 580 newslack 50 type g
status sb parent-gen 5 conf-gen 5 parent (5) children (13) cur-it 1

num 13 job carla res sco job-assign-st 555 job-in-conf yes num-conf 13
proptime 555 propslack 75 newtime 580 ftime 610 newslack 50 type cr
status new parent-gen 5 conf-gen 5 parent (5) no-children cur-it 1
```

```
job-propagation-after-conflict-resolution

num 5 job carla res fin job-assign-st 570 job-in-conf yes num-conf 5
proptime 545 propslack 75 newtime 570 ftime 580 newslack 50 type g
status sb parent-gen 5 conf-gen 5 parent (5) children (13) cur-it 1

num 13 job carla res sco job-assign-st 555 job-in-conf yes num-conf 13
proptime 555 propslack 75 newtime 580 ftime 610 newslack 50 type cr
status new parent-gen 5 conf-gen 5 parent (5) no-children cur-it 1
```

JTA Flavio

```
job-propagation-before-conflict-resolution

num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) children (6) cur-it 1

num 1 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 6
num-conf 1 proptime 590 propslack 30 newtime 595 ftime 600 newslack 25
type g status sb parent-gen 1 conf-gen 1 parent (1) children (9)
cur-it 1

num 8 job flavio res fin job-assign-st 600 job-in-conf yes num-conf 10
num-conf 8 proptime 600 propslack 30 newtime 615 ftime 675 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1

num 7 job flavio res eur job-assign-st 595 job-in-conf yes num-conf 9
num-conf 7 proptime 595 propslack 30 newtime 610 ftime 615 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (6) children (8)
cur-it 1

num 6 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 6
num-conf 1 proptime 590 propslack 30 newtime 605 ftime 610 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (2) children (7)
cur-it 1

num 10 job flavio res fin job-assign-st 600 job-in-conf yes num-conf
10 num-conf 8 proptime 600 propslack 30 newtime 605 ftime 665 newslack
25 type cr status new parent-gen 1 conf-gen 1 parent (9) no-children
cur-it 1

num 9 job flavio res eur job-assign-st 595 job-in-conf yes num-conf 9
num-conf 7 proptime 595 propslack 30 newtime 600 ftime 605 newslack 25
```

```
type cr status new parent-gen 1 conf-gen 1 parent (1) children (10)
cur-it 1


job-propagation-after-conflict-resolution

num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) children (6) cur-it 1

num 6 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 6
num-conf 1 proptime 590 propslack 30 newtime 605 ftime 610 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (2) children (7)
cur-it 1

num 7 job flavio res eur job-assign-st 595 job-in-conf yes num-conf 9
num-conf 7 proptime 595 propslack 30 newtime 610 ftime 615 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (6) children (8)
cur-it 1

num 8 job flavio res fin job-assign-st 600 job-in-conf yes num-conf 10
num-conf 8 proptime 600 propslack 30 newtime 615 ftime 675 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1
```

JTA Nelson

```
job-propagation-before-conflict-resolution

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 12
num-conf 4 proptime 584 propslack 66 newtime 600 ftime 610 newslack 50
type g status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it
1

num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 3
proptime 570 propslack 66 newtime 585 ftime 595 newslack 51 type g
status sb parent-gen 3 conf-gen 3 parent (3) children (11) cur-it 1

num 12 job nelson res gua job-assign-st 600 job-in-conf yes num-conf
12 num-conf 4 proptime 584 propslack 66 newtime 599 ftime 609 newslack
51 type cr status new parent-gen 3 conf-gen 3 parent (11) no-children
cur-it 1

num 11 job nelson res eur job-assign-st 580 job-in-conf yes num-conf
11 proptime 580 propslack 66 newtime 595 ftime 599 newslack 51 type cr
status new parent-gen 3 conf-gen 3 parent (3) children (12) cur-it 1
```

```
job-propagation-after-conflict-resolution

num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 3
proptime 570 propslack 66 newtime 585 ftime 595 newslack 51 type g
status sb parent-gen 3 conf-gen 3 parent (3) children (11) cur-it 1

num 11 job nelson res eur job-assign-st 580 job-in-conf yes num-conf
11 proptime 580 propslack 66 newtime 595 ftime 599 newslack 51 type cr
status new parent-gen 3 conf-gen 3 parent (3) no-children cur-it 1

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 12
num-conf 4 proptime 584 propslack 66 newtime 600 ftime 610 newslack 50
type g status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it
1
```

Plans

```
plan-with-lrs

num 5 job carla res fin job-assign-st 570 job-in-conf yes num-conf 5
proptime 545 propslack 75 newtime 570 ftime 580 newslack 50 type g
status sb parent-gen 5 conf-gen 5 parent (5) children (13) cur-it 1

num 13 job carla res sco job-assign-st 555 job-in-conf yes num-conf 13
proptime 555 propslack 75 newtime 580 ftime 610 newslack 50 type cr
status new parent-gen 5 conf-gen 5 parent (5) children (18 17 16 15
14) cur-it 1

num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 17
num-conf 3 proptime 570 propslack 66 newtime 585 ftime 595 newslack 51
type g status sb parent-gen 3 conf-gen 3 parent (3) children (11)
cur-it 1

num 17 job nelson res sco job-assign-st 585 job-in-conf yes num-conf
17 num-conf 3 proptime 570 propslack 66 newtime 570 ftime 580 newslack
66 type lr status new parent-gen 5 conf-gen 5 parent (13) no-children
cur-it 1

num 18 job ian res sco job-assign-st 570 job-in-conf yes num-conf 18
proptime 570 propslack 27 newtime 570 ftime 660 newslack 27 type lr
status new parent-gen 5 conf-gen 5 parent (13) no-children cur-it 1

num 11 job nelson res eur job-assign-st 580 job-in-conf yes num-conf
```

```
11 proptime 580 propslack 66 newtime 595 ftime 599 newslack 51 type
cr status new parent-gen 3 conf-gen 3 parent (3) children (22 21 20
19) cur-it 1

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 12
num-conf 4 proptime 584 propslack 66 newtime 600 ftime 610 newslack 50
type g status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it
1

num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) children (6) cur-it 1

num 1 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 16
num-conf 6 num-conf 1 proptime 590 propslack 30 newtime 595 ftime 600
newslack 25 type g status cancelled parent-gen 1 conf-gen 1 parent (1)
no-children cur-it 1

num 16 job flavio res sco job-assign-st 595 job-in-conf yes num-conf
16 num-conf 6 num-conf 1 proptime 590 propslack 30 newtime 590 ftime
595 newslack 30 type lr status new parent-gen 5 conf-gen 5 parent (13)
no-children cur-it 1

num 6 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 16
num-conf 6 num-conf 1 proptime 590 propslack 30 newtime 605 ftime 610
newslack 15 type cr status new parent-gen 2 conf-gen 2 parent (2)
children (24 23 7) cur-it 1

num 22 job flavio res eur job-assign-st 595 job-in-conf yes num-conf
22 num-conf 9 num-conf 7 proptime 595 propslack 30 newtime 595 ftime
600 newslack 30 type lr status new parent-gen 3 conf-gen 3 parent (11)
no-children cur-it 1

num 7 job flavio res eur job-assign-st 595 job-in-conf yes num-conf 22
num-conf 9 num-conf 7 proptime 595 propslack 30 newtime 610 ftime 615
newslack 15 type cr status new parent-gen 2 conf-gen 2 parent (6)
children (27 26 25 8) cur-it 1

num 20 job suresh res eur job-assign-st 600 job-in-conf yes num-conf
26 num-conf 20 proptime 600 propslack 2 newtime 600 ftime 603 newslack
2 type lr status new parent-gen 3 conf-gen 3 parent (11) no-children
cur-it 1

num 21 job alan res eur job-assign-st 600 job-in-conf yes num-conf 27
num-conf 21 proptime 600 propslack 23 newtime 600 ftime 602 newslack
23 type lr status new parent-gen 3 conf-gen 3 parent (11) no-children
```

```
cur-it 1

num 8 job flavio res fin job-assign-st 600 job-in-conf yes num-conf 10
num-conf 8 proptime 600 propslack 30 newtime 615 ftime 675 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (7) children (29 28)
cur-it 1

num 26 job suresh res eur job-assign-st 600 job-in-conf yes num-conf
26 num-conf 20 proptime 600 propslack 2 newtime 600 ftime 603 newslack
2 type lr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1

num 27 job alan res eur job-assign-st 600 job-in-conf yes num-conf 27
num-conf 21 proptime 600 propslack 23 newtime 600 ftime 602 newslack
23 type lr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1

num 15 job alan res sco job-assign-st 602 job-in-conf yes num-conf 24
num-conf 15 proptime 602 propslack 23 newtime 602 ftime 607 newslack
23 type lr status new parent-gen 5 conf-gen 5 parent (13) no-children
cur-it 1

num 24 job alan res sco job-assign-st 602 job-in-conf yes num-conf 24
num-conf 15 proptime 602 propslack 23 newtime 602 ftime 607 newslack
23 type lr status new parent-gen 2 conf-gen 2 parent (6) no-children
cur-it 1

num 29 job suresh res fin job-assign-st 603 job-in-conf yes num-conf
29 proptime 603 propslack 2 newtime 603 ftime 628 newslack 2 type lr
status new parent-gen 2 conf-gen 2 parent (8) no-children cur-it 1

num 14 job suresh res sco job-assign-st 628 job-in-conf yes num-conf
23 num-conf 14 proptime 628 propslack 2 newtime 628 ftime 638 newslack
2 type lr status new parent-gen 5 conf-gen 5 parent (13) no-children
cur-it 1

num 23 job suresh res sco job-assign-st 628 job-in-conf yes num-conf
23 num-conf 14 proptime 628 propslack 2 newtime 628 ftime 638 newslack
2 type lr status new parent-gen 2 conf-gen 2 parent (6) no-children
cur-it 1

num 28 job ian res fin job-assign-st 660 job-in-conf yes num-conf 28
proptime 660 propslack 27 newtime 660 ftime 661 newslack 27 type lr
status new parent-gen 2 conf-gen 2 parent (8) no-children cur-it 1

num 19 job ian res eur job-assign-st 662 job-in-conf yes num-conf 25
```

```
num-conf 19 proptime 662 propslack 27 newtime 662 ftime 663 newslack
27 type lr status new parent-gen 3 conf-gen 3 parent (11) no-children
cur-it 1

num 25 job ian res eur job-assign-st 662 job-in-conf yes num-conf 25
num-conf 19 proptime 662 propslack 27 newtime 662 ftime 663 newslack
27 type lr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1


plan-with-lrs-reds

num 5 job carla res fin job-assign-st 570 job-in-conf yes num-conf 5
proptime 545 propslack 75 newtime 570 ftime 580 newslack 50 type g
status sb parent-gen 5 conf-gen 5 parent (5) children (13) cur-it 1

num 13 job carla res sco job-assign-st 555 job-in-conf yes num-conf 13
proptime 555 propslack 75 newtime 580 ftime 610 newslack 50 type cr
status new parent-gen 5 conf-gen 5 parent (5) children (18 15 14)
cur-it 1

num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 17
num-conf 3 proptime 570 propslack 66 newtime 585 ftime 595 newslack 51
type cr status sb parent-gen 3 conf-gen 3 parent (3) children (11)
cur-it 1

num 18 job ian res sco job-assign-st 570 job-in-conf yes num-conf 18
proptime 570 propslack 27 newtime 570 ftime 660 newslack 27 type lr
status new parent-gen 5 conf-gen 5 parent (13) no-children cur-it 1

num 11 job nelson res eur job-assign-st 580 job-in-conf yes num-conf
11 proptime 580 propslack 66 newtime 595 ftime 599 newslack 51 type cr
status new parent-gen 3 conf-gen 3 parent (3) children (21 20 19)
cur-it 1

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 12
num-conf 4 proptime 584 propslack 66 newtime 600 ftime 610 newslack 50
type g status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it
1

num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) children (6) cur-it 1

num 6 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 16
num-conf 6 num-conf 1 proptime 590 propslack 30 newtime 605 ftime 610
```

```
newslack 15 type cr status new parent-gen 2 conf-gen 2 parent (2)
children (24 23 7) cur-it 1

num 7 job flavio res eur job-assign-st 595 job-in-conf yes num-conf 22
num-conf 9 num-conf 7 proptime 595 propslack 30 newtime 610 ftime 615
newslack 15 type cr status new parent-gen 2 conf-gen 2 parent (6)
children (27 26 25 8) cur-it 1

num 20 job suresh res eur job-assign-st 600 job-in-conf yes num-conf
26 num-conf 20 proptime 600 propslack 2 newtime 600 ftime 603 newslack
2 type lr status new parent-gen 3 conf-gen 3 parent (11) no-children
cur-it 1

num 21 job alan res eur job-assign-st 600 job-in-conf yes num-conf 27
num-conf 21 proptime 600 propslack 23 newtime 600 ftime 602 newslack
23 type lr status new parent-gen 3 conf-gen 3 parent (11) no-children
cur-it 1

num 8 job flavio res fin job-assign-st 600 job-in-conf yes num-conf 10
num-conf 8 proptime 600 propslack 30 newtime 615 ftime 675 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (7) children (29 28)
cur-it 1

num 26 job suresh res eur job-assign-st 600 job-in-conf yes num-conf
26 num-conf 20 proptime 600 propslack 2 newtime 600 ftime 603 newslack
2 type lr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1

num 27 job alan res eur job-assign-st 600 job-in-conf yes num-conf 27
num-conf 21 proptime 600 propslack 23 newtime 600 ftime 602 newslack
23 type lr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1

num 15 job alan res sco job-assign-st 602 job-in-conf yes num-conf 24
num-conf 15 proptime 602 propslack 23 newtime 602 ftime 607 newslack
23 type lr status new parent-gen 5 conf-gen 5 parent (13) no-children
cur-it 1

num 24 job alan res sco job-assign-st 602 job-in-conf yes num-conf 24
num-conf 15 proptime 602 propslack 23 newtime 602 ftime 607 newslack
23 type lr status new parent-gen 2 conf-gen 2 parent (6) no-children
cur-it 1

num 29 job suresh res fin job-assign-st 603 job-in-conf yes num-conf
29 proptime 603 propslack 2 newtime 603 ftime 628 newslack 2 type lr
status new parent-gen 2 conf-gen 2 parent (8) no-children cur-it 1
```

```
num 14 job suresh res sco job-assign-st 628 job-in-conf yes num-conf
23 num-conf 14 proptime 628 propslack 2 newtime 628 ftime 638 newslack
2 type lr status new parent-gen 5 conf-gen 5 parent (13) no-children
cur-it 1

num 23 job suresh res sco job-assign-st 628 job-in-conf yes num-conf
23 num-conf 14 proptime 628 propslack 2 newtime 628 ftime 638 newslack
2 type lr status new parent-gen 2 conf-gen 2 parent (6) no-children
cur-it 1

num 28 job ian res fin job-assign-st 660 job-in-conf yes num-conf 28
proptime 660 propslack 27 newtime 660 ftime 661 newslack 27 type lr
status new parent-gen 2 conf-gen 2 parent (8) no-children cur-it 1

num 19 job ian res eur job-assign-st 662 job-in-conf yes num-conf 25
num-conf 19 proptime 662 propslack 27 newtime 662 ftime 663 newslack
27 type lr status new parent-gen 3 conf-gen 3 parent (11) no-children
cur-it 1

num 25 job ian res eur job-assign-st 662 job-in-conf yes num-conf 25
num-conf 19 proptime 662 propslack 27 newtime 662 ftime 663 newslack
27 type lr status new parent-gen 2 conf-gen 2 parent (7) no-children
cur-it 1


final-plan

num 5 job carla res fin job-assign-st 570 job-in-conf yes num-conf 5
proptime 545 propslack 75 newtime 570 ftime 580 newslack 50 type g
status sb parent-gen 5 conf-gen 5 parent (5) children (13) cur-it 1

num 13 job carla res sco job-assign-st 555 job-in-conf yes num-conf 13
proptime 555 propslack 75 newtime 580 ftime 610 newslack 50 type cr
status sent parent-gen 5 conf-gen 5 parent (5) children (18 15 14)
cur-it 1

num 3 job nelson res sco job-assign-st 585 job-in-conf yes num-conf 17
num-conf 3 proptime 570 propslack 66 newtime 585 ftime 595 newslack 51
type cr status sent parent-gen 3 conf-gen 3 parent (3) children (11)
cur-it 1

num 18 job ian res sco job-assign-st 570 job-in-conf yes num-conf 18
proptime 570 propslack 27 newtime 570 ftime 660 newslack 27 type lr
status sent parent-gen 5 conf-gen 5 parent (13) no-children cur-it 1

num 11 job nelson res eur job-assign-st 580 job-in-conf yes num-conf
```

```
11 proptime 580 propslack 66 newtime 595 ftime 599 newslack 51 type cr
status new parent-gen 3 conf-gen 3 parent (3) children (21 20 19)
cur-it 1

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 12
num-conf 4 proptime 584 propslack 66 newtime 600 ftime 610 newslack 50
type g status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it
1

num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) children (6) cur-it 1

num 6 job flavio res sco job-assign-st 595 job-in-conf yes num-conf 16
num-conf 6 num-conf 1 proptime 590 propslack 30 newtime 605 ftime 610
newslack 15 type cr status sent parent-gen 2 conf-gen 2 parent (2)
children (14 15 7) cur-it 1

num 7 job flavio res eur job-assign-st 595 job-in-conf yes num-conf 22
num-conf 9 num-conf 7 proptime 595 propslack 30 newtime 610 ftime 615
newslack 15 type cr status new parent-gen 2 conf-gen 2 parent (6)
children (19 21 20 8) cur-it 1

num 8 job flavio res fin job-assign-st 600 job-in-conf yes num-conf 10
num-conf 8 proptime 600 propslack 30 newtime 615 ftime 675 newslack 15
type cr status new parent-gen 2 conf-gen 2 parent (7) children (29 28)
cur-it 1

num 20 job suresh res eur job-assign-st 600 job-in-conf yes num-conf
26 num-conf 20 proptime 600 propslack 2 newtime 600 ftime 603 newslack
2 type lr status new parent-gen 3 conf-gen 3 parent (7 11) no-children
cur-it 1

num 21 job alan res eur job-assign-st 600 job-in-conf yes num-conf 27
num-conf 21 proptime 600 propslack 23 newtime 600 ftime 602 newslack
23 type lr status new parent-gen 3 conf-gen 3 parent (7 11)
no-children cur-it 1

num 15 job alan res sco job-assign-st 602 job-in-conf yes num-conf 24
num-conf 15 proptime 602 propslack 23 newtime 602 ftime 607 newslack
23 type lr status sent parent-gen 5 conf-gen 5 parent (6 13)
no-children cur-it 1

num 29 job suresh res fin job-assign-st 603 job-in-conf yes num-conf
29 proptime 603 propslack 2 newtime 603 ftime 628 newslack 2 type lr
status new parent-gen 2 conf-gen 2 parent (8) no-children cur-it 1
```

```
num 14 job suresh res sco job-assign-st 628 job-in-conf yes num-conf
23 num-conf 14 proptime 628 propslack 2 newtime 628 ftime 638 newslack
2 type lr status sent parent-gen 5 conf-gen 5 parent (6 13)
no-children cur-it 1

num 28 job ian res fin job-assign-st 660 job-in-conf yes num-conf 28
proptime 660 propslack 27 newtime 660 ftime 661 newslack 27 type lr
status new parent-gen 2 conf-gen 2 parent (8) no-children cur-it 1

num 19 job ian res eur job-assign-st 662 job-in-conf yes num-conf 25
num-conf 19 proptime 662 propslack 27 newtime 662 ftime 663 newslack
27 type lr status new parent-gen 3 conf-gen 3 parent (7 11)
no-children cur-it 1
```

B.2 Iteration 3

Conflict Propagation

JTA Alan

```
job-propagation-after-conflict-resolution

num 31 job alan res sco job-assign-st 615 job-in-conf yes num-conf 31
num-conf 24 num-conf 15 proptime 602 propslack 23 newtime 615 ftime
620 newslack 10 type g status sb parent-gen 5 conf-gen 31 parent (31)
no-children cur-it 2
```

JTA Ian

```
job-propagation-after-conflict-resolution

num 32 job ian res sco job-assign-st 595 job-in-conf yes num-conf 32
num-conf 18 proptime 570 propslack 27 newtime 595 ftime 685 newslack 2
type g status sb parent-gen 5 conf-gen 32 parent (32) children (35)
cur-it 2

num 35 job ian res fin job-assign-st 660 job-in-conf yes num-conf 35
num-conf 28 proptime 660 propslack 27 newtime 685 ftime 686 newslack 2
type cr status new parent-gen 5 conf-gen 32 parent (32) children (36)
cur-it 2

num 36 job ian res gua job-assign-st 661 job-in-conf yes num-conf 36
```

```
proptime 661 propslack 27 newtime 686 ftime 687 newslack 2 type cr
status new parent-gen 5 conf-gen 32 parent (35) children (37) cur-it 2

num 37 job ian res eur job-assign-st 662 job-in-conf yes num-conf 37
num-conf 25 num-conf 19 proptime 662 propslack 27 newtime 687 ftime
688 newslack 2 type cr status new parent-gen 5 conf-gen 32 parent (36)
no-children cur-it 2
```

JTA Flavio

```
job-propagation-after-conflict-resolution

num 30 job flavio res sco job-assign-st 610 job-in-conf yes num-conf
30 num-conf 16 num-conf 6 num-conf 1 proptime 605 propslack 15 newtime
610 ftime 615 newslack 10 type g status sb parent-gen 2 conf-gen 30
parent (30) children (33) cur-it 2

num 33 job flavio res eur job-assign-st 595 job-in-conf yes num-conf
33 num-conf 22 num-conf 9 num-conf 7 proptime 595 propslack 30 newtime
615 ftime 620 newslack 10 type cr status new parent-gen 2 conf-gen 30
parent (30) children (34) cur-it 2

num 34 job flavio res fin job-assign-st 600 job-in-conf yes num-conf
34 num-conf 10 num-conf 8 proptime 600 propslack 30 newtime 620 ftime
680 newslack 10 type cr status new parent-gen 2 conf-gen 30 parent
(33) no-children cur-it 2
```

Plans

```
final-plan

num 32 job ian res sco job-assign-st 595 job-in-conf yes num-conf 32
num-conf 18 proptime 570 propslack 27 newtime 595 ftime 685 newslack 2
type g status sb parent-gen 5 conf-gen 32 parent (32) children (35)
cur-it 2

num 11 job nelson res eur job-assign-st 580 job-in-conf yes num-conf
11 proptime 580 propslack 66 newtime 595 ftime 599 newslack 51 type cr
status sent parent-gen 3 conf-gen 3 parent (3) no-children cur-it 1

num 4 job nelson res gua job-assign-st 600 job-in-conf yes num-conf 12
num-conf 4 proptime 584 propslack 66 newtime 600 ftime 610 newslack 50
type g status sb parent-gen 4 conf-gen 4 parent (4) no-children cur-it
1
```

```
num 2 job flavio res gua job-assign-st 600 job-in-conf yes num-conf 2
proptime 585 propslack 30 newtime 600 ftime 605 newslack 15 type g
status sb parent-gen 2 conf-gen 2 parent (2) no-children cur-it 1

num 33 job flavio res eur job-assign-st 595 job-in-conf yes num-conf
33 num-conf 22 num-conf 9 num-conf 7 proptime 595 propslack 30 newtime
615 ftime 620 newslack 10 type cr status sent parent-gen 2 conf-gen 30
parent (30) children (40 39 34) cur-it 2

num 34 job flavio res fin job-assign-st 600 job-in-conf yes num-conf
34 num-conf 10 num-conf 8 proptime 600 propslack 30 newtime 620 ftime
680 newslack 10 type cr status new parent-gen 2 conf-gen 30 parent
(33) children (42) cur-it 2

num 39 job suresh res eur job-assign-st 600 job-in-conf yes num-conf
39 num-conf 26 num-conf 20 proptime 600 propslack 2 newtime 600 ftime
603 newslack 2 type lr status sent parent-gen 2 conf-gen 30 parent
(33) no-children cur-it 2

num 40 job alan res eur job-assign-st 600 job-in-conf yes num-conf 40
num-conf 27 num-conf 21 proptime 600 propslack 23 newtime 600 ftime
602 newslack 23 type lr status sent parent-gen 2 conf-gen 30 parent
(33) no-children cur-it 2

num 31 job alan res sco job-assign-st 615 job-in-conf yes num-conf 31
num-conf 24 num-conf 15 proptime 602 propslack 23 newtime 615 ftime
620 newslack 10 type g status sb parent-gen 5 conf-gen 31 parent (31)
no-children cur-it 2

num 42 job suresh res fin job-assign-st 603 job-in-conf yes num-conf
42 num-conf 29 proptime 603 propslack 2 newtime 603 ftime 628 newslack
2 type lr status new parent-gen 2 conf-gen 30 parent (34) no-children
cur-it 2

num 30 job flavio res sco job-assign-st 610 job-in-conf yes num-conf
30 num-conf 16 num-conf 6 num-conf 1 proptime 605 propslack 15 newtime
610 ftime 615 newslack 10 type g status sb parent-gen 2 conf-gen 30
parent (30) children (33) cur-it 2

num 35 job ian res fin job-assign-st 660 job-in-conf yes num-conf 41
num-conf 35 num-conf 28 proptime 660 propslack 27 newtime 685 ftime
686 newslack 2 type cr status new parent-gen 5 conf-gen 32 parent (32)
children (36) cur-it 2

num 36 job ian res gua job-assign-st 661 job-in-conf yes num-conf 36
proptime 661 propslack 27 newtime 686 ftime 687 newslack 2 type cr
```

```
status new parent-gen 5 conf-gen 32 parent (35) children (37) cur-it 2

num 37 job ian res eur job-assign-st 662 job-in-conf yes num-conf 38
num-conf 37 num-conf 25 num-conf 19 proptime 662 propslack 27 newtime
687 ftime 688 newslack 2 type cr status sent parent-gen 5 conf-gen 32
parent (36) no-children cur-it 2
```

B.3 Iteration 4

Plans

```
final-plan

num 32 job ian res sco job-assign-st 595 job-in-conf yes num-conf 32
num-conf 18 proptime 570 propslack 27 newtime 595 ftime 685 newslack 2
type g status sb parent-gen 5 conf-gen 32 parent (32) children (35)
cur-it 2

num 34 job flavio res fin job-assign-st 600 job-in-conf yes num-conf
34 num-conf 10 num-conf 8 proptime 600 propslack 30 newtime 620 ftime
680 newslack 10 type cr status sent parent-gen 2 conf-gen 30 parent
(30) children (42) cur-it 2

num 31 job alan res sco job-assign-st 615 job-in-conf yes num-conf 31
num-conf 24 num-conf 15 proptime 602 propslack 23 newtime 615 ftime
620 newslack 10 type g status sb parent-gen 5 conf-gen 31 parent (31)
no-children cur-it 2

num 42 job suresh res fin job-assign-st 603 job-in-conf yes num-conf
42 num-conf 29 proptime 603 propslack 2 newtime 603 ftime 628 newslack
2 type lr status sent parent-gen 2 conf-gen 30 parent (34) no-children
cur-it 2

num 30 job flavio res sco job-assign-st 610 job-in-conf yes num-conf
30 num-conf 16 num-conf 6 num-conf 1 proptime 605 propslack 15 newtime
610 ftime 615 newslack 10 type g status sb parent-gen 2 conf-gen 30
parent (30) children (33 40 39 34) cur-it 2

num 35 job ian res fin job-assign-st 660 job-in-conf yes num-conf 41
num-conf 35 num-conf 28 proptime 660 propslack 27 newtime 685 ftime
686 newslack 2 type cr status sent parent-gen 5 conf-gen 32 parent
(32) children (36) cur-it 2

num 36 job ian res gua job-assign-st 661 job-in-conf yes num-conf 36
proptime 661 propslack 27 newtime 686 ftime 687 newslack 2 type cr
```

```
status new parent-gen 5 conf-gen 32 parent (35) children (37) cur-it 2
```

B.4 Iteration 5

Plans

```
final-plan

num 32 job ian res sco job-assign-st 595 job-in-conf yes num-conf 32
num-conf 18 proptime 570 propslack 27 newtime 595 ftime 685 newslack 2
type g status sb parent-gen 5 conf-gen 32 parent (32) children (35 36)
cur-it 2

num 36 job ian res gua job-assign-st 661 job-in-conf yes num-conf 36
proptime 661 propslack 27 newtime 686 ftime 687 newslack 2 type cr
status sent parent-gen 5 conf-gen 32 parent (32) children (37) cur-it
2
```

Appendix C

Main Features of The Test Data

This appendix summarises the main characteristics of the data used to test the performance of EXPLICIT, generated according to the procedure described in chapter 9.

Cases	*Number of Jobs*	*Number of Machines per Agg. Resource*
1	6	2
2	7	2
3	6	2
4	7	2
5	8	2
6	8	2
7	11	2
8	13	2
9	12	2
10	20	2
11	18	2
12	15	2
13	22	3
14	24	3
15	24	3
16	27	5
17	29	5
18	29	5

Table C.1: The cases for type A jobs

Cases	*Number of Jobs*	*Number of Machines per Agg. Resource*
19	6	2
20	7	2
21	6	2
22	9	2
23	8	2
24	8	2
25	13	2
26	15	2
27	12	2
28	20	2
29	18	2
30	15	2
31	23	3
32	20	3
33	23	3
34	27	5
35	29	5
36	29	5

Table C.2: The cases for type B jobs

Cases	*Number of Jobs*	*Number of Machines per Agg. Resource*
37	7	2
38	6	2
39	7	2
40	9	2
41	9	2
42	9	2
43	10	2
44	11	2
45	12	2
46	17	2
47	15	2
48	17	2
49	21	3
50	22	3
51	20	3
52	27	5
53	29	5
54	26	5

Table C.3: The cases for the type C jobs

Appendix D

Results - Performance Measures

In this appendix the results in terms of the performance measures discussed in chapter 9 are listed for both EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN. This appendix complements the discussion presented in that chapter in terms of the analysis of the performance of the system.

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
1	790	691.50	256	-156	0	0
2	714	670.57	185	-175	0	0
3	717	644.17	184	-171	0	0
4	755	692.29	248	-122	0	0
5	763	666.50	235	-136	0	0
6	821	715.38	284	-108	0	0
7	807	716.18	278	-84	0	0
8	956	775.15	419	27	27	1
9	860	735.00	343	-31	0	0
10	1027	835.35	488	127	127	10
11	1081	842.11	544	134	134	8
12	1045	813.60	527	94	94	5
13	959	755.41	421	18	18	1
14	1028	805.88	514	147	147	9
15	961	810.88	446	67	67	8
16	864	701.59	326	-77	0	0
17	961	763.52	428	31	31	2
18	815	704.69	327	-55	0	0

Table D.1: Jobs perfomance measures for EXPLICIT-OAS-OUT

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
19	717	645.50	183	-133	0	0
20	712	645.00	183	-141	0	0
21	725	622.17	192	-151	0	0
22	801	654.00	300	-64	0	0
23	735	666.25	209	-100	0	0
24	774	661.88	257	-135	0	0
25	873	716.46	350	-7	0	0
26	946	777.27	444	50	50	6
27	788	680.92	264	-48	0	0
28	1123	858.20	595	234	234	13
29	1128	851.00	591	241	241	9
30	916	736.87	398	45	45	2
31	917	743.17	402	36	36	4
32	863	712.00	362	23	23	2
33	1016	808.91	488	171	171	11
34	815	683.19	289	-65	0	0
35	829	685.41	316	-13	0	0
36	844	704.07	343	-13	0	0
37	715	636.29	219	-113	0	0
38	711	658.50	185	-136	0	0
39	676	638.29	163	-128	0	0
40	728	618.56	223	-99	0	0
41	757	662.78	237	-79	0	0
42	799	635.67	284	-104	0	0
43	810	698.30	275	-52	0	0
44	885	701.64	349	-4	0	0
45	790	678.50	270	-25	0	0
46	960	744.00	443	111	111	6
47	898	700.67	369	16	16	1
48	975	745.88	461	109	109	5
49	927	725.33	413	61	61	4
50	937	720.18	423	71	71	4
51	828	682.45	314	18	18	1
52	868	705.48	335	28	28	2
53	840	669.00	303	-43	0	0
54	805	664.92	303	-62	0	0

Table D.2: Jobs perfomance measures for EXPLICIT-OAS-OUT (cont)

Case	*Mean Idle Time per Machine*
1	78.18
2	40.42
3	34.45
4	54.00
5	50.67
6	93.73
7	69.58
8	143.58
9	143.73
10	174.25
11	176.25
12	200.33
13	128.29
14	181.67
15	143.18
16	62.85
17	141.04
18	68.56
19	47.64
20	41.08
21	45.08
22	81.83
23	65.42
24	47.08
25	119.33
26	152.25
27	67.17

Table D.3: Resources Perfomance Measures for EXPLICIT-OAS-OUT

Case	*Mean Idle Time per Machine*
28	144.17
29	216.67
30	125.75
31	135.83
32	96.56
33	172.11
34	59.38
35	73.00
36	68.54
37	31.55
38	54.92
39	31.17
40	63.83
41	66.58
42	85.08
43	93.58
44	83.00
45	44.50
46	95.58
47	96.17
48	131.83
49	82.17
50	98.89
51	52.72
52	76.22
53	44.50
54	39.34

Table D.4: Resources Perfomance Measures forEXPLICIT-OAS-OUT (cont.)

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
1	790	691.50	256	-156	0	0
2	714	670.57	185	-175	0	0
3	717	644.17	184	-171	0	0
4	788	689.29	281	-143	0	0
5	785	674.00	251	-114	0	0
6	848	711.88	314	-108	0	0
7	763	706.91	256	-107	0	0
8	956	778.92	419	27	27	1
9	821	719.75	304	-77	0	0
10	1025	811.75	486	125	125	6
11	1047	817.50	510	100	100	6
12	1030	801.80	512	92	92	5
13	957	743.36	419	16	16	1
14	1008	791.46	496	101	101	6
15	956	795.58	441	49	49	7
16	849	694.67	315	-102	0	0
17	927	751.03	394	-3	0	0
18	868	725.72	354	19	19	2
19	717	645.50	183	-133	0	0
20	711	640.00	182	-152	0	0
21	725	622.17	192	-151	0	0
22	828	684.44	310	21	21	1
23	718	651.38	209	-105	0	0
24	774	661.88	257	-135	0	0
25	883	715.77	353	3	3	1
26	944	751.13	442	48	48	5
27	782	670.08	258	-54	0	0

Table D.5: Jobs perfomance measures for EXPLICIT-OAS-IN

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
28	1065	834.10	531	278	278	11
29	1039	792.39	542	184	184	6
30	959	751.80	441	88	88	4
31	917	725.57	412	36	36	3
32	874	708.25	377	-1	0	0
33	1006	759.39	467	123	123	8
34	815	674.56	289	-74	0	0
35	854	685.59	324	65	65	2
36	844	702.79	343	-13	0	0
37	715	647.57	206	-117	0	0
38	711	658.50	185	-136	0	0
39	676	638.57	171	-129	0	0
40	756	644.11	223	-59	0	0
41	729	654.89	237	-88	0	0
42	791	631.56	276	-112	0	0
43	760	674.30	227	-88	0	0
44	869	697.45	333	-16	0	0
45	772	669.83	258	-38	0	0
46	914	717.53	392	51	51	3
47	898	704.20	369	16	16	1
48	1051	782.00	537	185	185	6
49	927	734.43	413	61	61	4
50	935	729.45	421	69	69	4
51	797	681.30	262	-18	0	0
52	826	688.85	298	-25	0	0
53	840	675.79	303	-43	0	0
54	805	658.08	303	-66	0	0

Table D.6: Jobs perfomance measures for EXPLICIT-OAS-IN (cont)

Case	*Mean Idle Time per Machine*
1	78.18
2	40.42
3	34.45
4	61.17
5	54.25
6	104.27
7	54.17
8	122.33
9	92.50
10	140.00
11	161.17
12	170.67
13	123.12
14	173.17
15	111.61
16	58.07
17	136.48
18	83.22
19	47.64
20	33.92
21	45.08
22	117.67
23	46.92
24	47.08
25	116.08
26	147.25
27	57.33

Table D.7: Resources Perfomance Measures for EXPLICIT-OAS-IN

Case	*Mean Idle Time per Machine*
28	137.67
29	162.50
30	141.08
31	134.50
32	75.56
33	102.28
34	47.21
35	62.21
36	62.39
37	35.00
38	54.92
39	34.42
40	62.08
41	43.00
42	80.83
43	63.83
44	95.33
45	43.25
46	79.50
47	96.92
48	184.00
49	100.50
50	112.39
51	56.33
52	65.37
53	54.25
54	30.52

Table D.8: Resources Perfomance Measures for EXPLICIT-OAS-IN (cont.)

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
1	844	705.83	310	-132	0	0
2	730	672.00	201	-163	0	0
3	727	653.00	194	-130	0	0
4	852	685.43	345	-93	0	0
5	848	685.50	328	-75	0	0
6	902	726.63	368	-74	0	0
7	889	704.55	382	-56	0	0
8	1042	784.46	508	66	66	1
9	958	735.33	444	9	9	1
10	1064	839.50	534	162	162	10
11	1116	827.61	583	148	148	8
12	1081	818.13	563	139	139	4
13	1014	741.09	480	38	38	1
14	1058	793.75	545	147	147	7
15	1031	788.08	520	116	116	6
16	935	705.15	401	-41	0	0
17	992	767.48	479	63	63	8
18	848	697.45	312	-59	0	0
19	708	645.50	208	-113	0	0
20	712	645.00	183	-141	0	0
21	672	614.17	142	-154	0	0
22	811	651.33	310	-83	0	0
23	739	660.88	230	-84	0	0
24	774	661.88	257	-135	0	0
25	926	718.69	408	14	14	1
26	949	763.60	428	73	73	6
27	803	683.83	279	-45	0	0

Table D.9: Jobs perfomance measures for MINSLK

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
28	1083	846.35	555	207	207	11
29	1046	798.78	556	187	187	8
30	942	743.87	424	71	71	4
31	960	739.96	475	84	84	4
32	867	698.70	370	-8	0	0
33	980	777.35	497	146	146	11
34	810	677.67	284	-62	0	0
35	822	675.55	309	-26	0	0
36	890	695.69	389	-2	0	0
37	734	639.86	225	-99	0	0
38	725	659.83	199	-128	0	0
39	704	644.14	219	-104	0	0
40	697	606.56	176	-132	0	0
41	777	653.89	285	-54	0	0
42	798	648.33	283	-81	0	0
43	790	680.60	255	-72	0	0
44	888	707.36	352	-1	0	0
45	802	655.17	282	-23	0	0
46	944	724.12	422	45	45	5
47	910	713.93	381	28	28	1
48	1008	751.76	514	160	160	7
49	927	728.52	433	79	79	5
50	927	725.59	433	79	79	5
51	810	671.75	290	16	16	1
52	850	684.15	356	2	2	1
53	846	677.59	309	-24	0	0
54	809	659.85	307	-31	0	0

Table D.10: Jobs perfomance measures for MINSLK (cont)

Case	*Mean Idle Time per Machine*
1	106.30
2	45.17
3	43.73
4	93.33
5	105.33
6	105.92
7	112.75
8	169.75
9	156.17
10	219.00
11	221.67
12	245.17
13	113.39
14	214.39
15	172.61
16	78.96
17	136.65
18	68.18
19	36.92
20	45.33
21	25.83
22	75.83
23	54.58
24	50.17
25	132.42
26	137.00
27	70.83

Table D.11: Resources Perfomance Measures for MINSLK

Case	*Mean Idle Time per Machine*
28	163.75
29	199.50
30	134.00
31	121.17
32	101.12
33	123.22
34	55.21
35	63.76
36	66.90
37	35.18
38	73.75
39	64.58
40	43.33
41	54.00
42	84.75
43	61.17
44	77.33
45	38.50
46	86.92
47	103.83
48	113.17
49	86.11
50	100.11
51	48.44
52	49.50
53	64.67
54	36.97

Table D.12: Resources Perfomance Measures for MINSLK (cont.)

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
1	781	688.83	277	-77	0	0
2	873	700.43	344	-18	0	0
3	710	641.33	177	-141	0	0
4	851	690.29	322	-40	0	0
5	759	676.00	225	-110	0	0
6	839	714.00	335	-19	0	0
7	860	689.91	331	-31	0	0
8	921	778.15	430	128	128	4
9	887	762.67	383	88	88	3
10	1089	801.80	588	214	214	8
11	1062	832.89	573	243	243	7
12	1008	797.27	516	192	192	5
13	937	770.73	433	129	129	8
14	1032	811.38	546	259	259	12
15	1053	802.75	546	238	238	9
16	856	713.00	352	19	19	2
17	963	783.07	465	178	178	10
18	890	709.24	383	75	75	4
19	744	649.83	240	-25	0	0
20	759	651.57	225	-53	0	0
21	672	610.50	139	-127	0	0
22	764	687.67	264	58	58	3
23	747	668.88	228	-47	0	0
24	811	694.75	273	9	9	2
25	851	718.62	343	75	75	4
26	968	760.60	452	141	141	6
27	803	687.08	301	13	13	1

Table D.13: Jobs perfomance measures for MAXSLK

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
28	1097	837.30	596	334	334	10
29	1093	871.11	582	346	346	10
30	885	749.87	380	143	143	5
31	932	748.83	433	185	185	8
32	889	735.15	395	172	172	6
33	1010	769.09	512	232	232	11
34	829	691.07	325	60	60	4
35	803	680.34	292	52	52	6
36	837	706.03	330	106	106	7
37	733	665.14	212	-15	0	0
38	765	650.83	245	-52	0	0
39	773	648.86	282	1	1	1
40	700	639.11	208	-24	0	0
41	731	663.00	245	11	11	1
42	814	661.89	319	8	8	1
43	801	701.30	290	73	73	2
44	854	722.55	367	165	165	4
45	816	699.42	331	106	106	4
46	950	760.29	450	227	227	7
47	919	744.20	424	202	202	7
48	1003	795.18	483	269	269	8
49	940	760.38	420	206	206	8
50	940	764.23	420	206	206	9
51	815	683.15	332	91	91	4
52	836	704.41	332	119	119	7
53	829	688.83	326	65	65	6
54	837	672.46	330	58	58	4

Table D.14: Jobs perfomance measures for MAXSLK (cont)

Case	*Mean Idle Time per Machine*
1	59.50
2	106.58
3	37.00
4	92.08
5	44.33
6	67.75
7	112.92
8	138.33
9	126.67
10	199.75
11	213.75
12	164.92
13	150.33
14	191.56
15	199.06
16	56.10
17	126.21
18	97.82
19	65.00
20	38.00
21	31.58
22	91.42
23	54.33
24	77.92
25	108.00
26	155.83
27	96.83

Table D.15: Resources Perfomance Measures for MAXSLK

Case	*Mean Idle Time per Machine*
28	234.42
29	224.25
30	123.58
31	156.11
32	130.67
33	96.28
34	62.38
35	60.34
36	65.59
37	72.17
38	37.08
39	82.92
40	36.25
41	57.33
42	92.83
43	67.92
44	111.17
45	151.25
46	146.42
47	162.25
48	158.42
49	127.22
50	142.72
51	83.50
52	83.63
53	76.79
54	52.34

Table D.16: Resources Perfomance Measures for MAXSLK (cont.)

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
1	842	688.00	308	-134	0	0
2	840	689.14	311	-51	0	0
3	710	648.17	177	-141	0	0
4	897	709.29	368	6	6	1
5	779	680.00	272	-127	0	0
6	900	705.50	366	-76	0	0
7	917	723.27	388	26	26	1
8	1034	768.00	500	58	58	3
9	927	732.25	413	-22	0	0
10	1144	791.10	643	269	269	6
11	1179	838.67	646	288	288	7
12	1084	809.40	569	228	228	6
13	1048	765.82	514	72	72	6
14	1094	785.08	580	165	165	6
15	1087	803.08	576	239	239	8
16	935	712.74	401	-5	0	0
17	993	744.21	479	73	73	5
18	892	699.52	375	67	67	3
19	744	646.83	240	-25	0	0
20	714	648.57	185	-140	0	0
21	673	607.33	140	-164	0	0
22	820	664.44	319	-64	0	0
23	828	669.50	321	-21	0	0
24	785	671.88	268	-40	0	0
25	885	711.85	369	55	55	2
26	1016	753.00	497	186	186	3
27	854	679.00	350	25	25	2

Table D.17: Jobs perfomance measures for SOF

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
28	1150	847.75	636	351	351	10
29	1135	820.56	645	268	268	8
30	999	733.40	502	141	141	3
31	1014	736.00	484	153	153	6
32	923	719.20	419	50	50	5
33	1025	774.39	511	230	230	10
34	863	683.11	337	54	54	2
35	825	677.86	299	5	5	1
36	921	704.83	420	84	84	4
37	742	653.29	254	-78	0	0
38	770	659.17	250	-47	0	0
39	775	647.43	295	-30	0	0
40	725	621.44	225	-81	0	0
41	790	654.89	305	-12	0	0
42	801	653.67	290	-21	0	0
43	867	698.00	345	-7	0	0
44	874	703.64	370	96	96	1
45	846	683.08	350	45	45	1
46	974	736.82	452	111	111	4
47	916	708.60	389	39	39	1
48	1002	756.94	486	201	201	5
49	940	728.57	424	139	139	5
50	940	729.45	424	139	139	6
51	847	676.20	357	46	46	2
52	852	682.67	336	51	51	2
53	860	680.72	354	22	22	1
54	823	667.81	316	12	12	1

Table D.18: Jobs perfomance measures for SOF (cont)

Case	*Mean Idle Time per Machine*
1	98.18
2	90.42
3	37.73
4	111.75
5	65.58
6	125.64
7	157.42
8	172.58
9	155.42
10	226.75
11	197.42
12	211.42
13	157.78
14	198.11
15	208.11
16	88.04
17	116.44
18	84.32
19	66.27
20	37.75
21	25.58
22	80.75
23	100.64
24	72.33
25	109.67
26	176.82
27	90.08

Table D.19: Resources Perfomance Measures for SOF

Case	*Mean Idle Time per Machine*
28	235.42
29	267.67
30	160.33
31	98.56
32	127.71
33	116.94
34	58.03
35	67.53
36	83.96
37	37.73
38	46.67
39	82.17
40	42.08
41	43.75
42	115.25
43	128.08
44	90.58
45	74.58
46	100.42
47	92.92
48	125.50
49	84.72
50	109.33
51	62.61
52	59.36
53	69.40
54	51.60

Table D.20: Resources Perfomance Measures for SOF (cont.)

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
1	715	700.00	207	-79	0	0
2	749	694.86	220	-75	0	0
3	710	657.33	179	-115	0	0
4	750	699.00	221	-105	0	0
5	753	661.13	219	-146	0	0
6	776	724.75	269	-34	0	0
7	799	716.91	270	-60	0	0
8	924	790.00	402	121	121	3
9	836	733.58	319	-10	0	0
10	1015	826.40	485	130	130	9
11	1097	853.33	560	225	225	8
12	1022	815.93	510	132	132	6
13	899	773.50	399	113	113	6
14	1019	826.71	537	218	218	9
15	1005	783.29	512	163	163	8
16	838	708.93	314	13	13	1
17	946	770.72	464	145	145	8
18	847	694.79	354	5	5	1
19	698	652.50	198	-123	0	0
20	714	648.57	185	-140	0	0
21	672	611.83	139	-145	0	0
22	748	650.11	247	-74	0	0
23	828	669.50	321	-21	0	0
24	785	671.88	268	-40	0	0
25	885	711.85	369	55	55	2
26	1016	753.00	497	186	186	3
27	795	690.50	271	-8	0	0

Table D.21: Jobs perfomance measures for MOF

Case	*MaxCT*	*MeanCT*	*MaxFT*	*MaxLat*	*MaxTard*	*Tardy Jobs*
28	1069	834.30	541	208	208	11
29	1046	843.17	533	252	252	10
30	900	745.60	375	53	53	5
31	934	753.22	449	153	153	7
32	878	734.25	354	98	98	5
33	1011	765.00	528	150	150	7
34	828	680.78	302	-8	0	0
35	814	665.21	288	-6	0	0
36	842	702.45	326	59	59	2
37	724	645.57	236	-96	0	0
38	731	651.67	209	-106	0	0
39	691	630.29	203	-83	0	0
40	733	612.89	233	-75	0	0
41	721	651.00	210	-83	0	0
42	739	673.44	218	-4	0	0
43	788	692.80	270	-23	0	0
44	887	722.64	351	30	30	4
45	784	670.42	256	47	47	1
46	930	723.65	408	120	120	5
47	911	748.47	375	128	128	5
48	978	787.59	491	278	278	8
49	928	765.24	441	228	228	8
50	940	767.86	453	240	240	9
51	776	668.00	280	15	15	1
52	830	701.96	339	126	126	4
53	820	696.45	313	49	49	5
54	766	666.46	285	4	4	1

Table D.22: Jobs perfomance measures for MOF (cont)

Case	*Mean Idle Time per Machine*
1	60.58
2	60.58
3	47.91
4	46.00
5	39.67
6	59.00
7	92.83
8	100.25
9	105.67
10	139.67
11	211.50
12	175.58
13	105.61
14	169.44
15	109.33
16	58.57
17	115.85
18	64.71
19	39.64
20	37.75
21	29.25
22	66.83
23	100.64
24	72.33
25	109.67
26	176.82
27	65.75

Table D.23: Resources Perfomance Measures for MOF

Case	*Mean Idle Time per Machine*
28	128.92
29	166.58
30	89.33
31	105.28
32	84.56
33	104.39
34	48.23
35	54.59
36	54.57
37	33.58
38	35.75
39	29.58
40	34.42
41	36.00
42	62.33
43	28.75
44	85.75
45	44.00
46	83.33
47	88.75
48	133.58
49	99.44
50	116.56
51	31.83
52	56.45
53	48.40
54	33.17

Table D.24: Resources Perfomance Measures for MOF (cont.)

Appendix E

Results - Iterations and Times

In this appendix the results in terms of number of iterations and cpu times are listed for both EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN. This appendix complements the discussion presented in chapter 9 regarding the analysis of the time perfomance of the system.

Iterations

Type A			
Number of Jobs	*Machines per Agg. Res.*	*Number of Iterations*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
06	2	06	06
06	2	06	07
07	2	06	06
07	2	07	07
08	2	06	06
08	2	06	10
11	2	08	06
12	2	10	06
13	2	06	06

Table E.1: The number of iterations (type A problems)

Type A			
Number of Jobs	*Machines per Agg. Res.*	*Number of Iterations*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
15	2	09	07
18	2	11	10
20	2	10	09
22	3	07	08
24	3	07	07
24	3	10	07
27	5	06	07
29	5	07	06
29	5	08	12

Table E.2: The number of iterations (type A problems) (cont.)

Type B			
Number of Jobs	*Machines per Agg. Res.*	*Number of Iterations*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
06	2	05	05
06	2	07	07
07	2	06	06
08	2	06	06
08	2	06	06
09	2	07	07
12	2	06	08
13	2	06	07
15	2	06	06
15	2	09	15
18	2	09	06
20	2	07	09
20	3	08	11
23	3	06	07
23	3	09	10
27	5	07	09
29	5	07	08
29	5	07	08

Table E.3: The number of iterations (type B problems)

Type C			
Number of Jobs	*Machines per Agg. Res.*	*Number of Iterations*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
06	2	06	06
07	2	04	05
07	2	05	07
09	2	06	06
09	2	06	08
09	2	08	09
10	2	07	06
11	2	06	06
12	2	07	09
15	2	06	09
17	2	07	08
17	2	08	13
20	3	07	10
21	3	07	08
22	3	06	06
26	5	08	07
27	5	08	08
29	5	06	08

Table E.4: The number of iterations (type C problems)

CPU Times

Type A			
Number of Jobs	*Machines per Agg. Res.*	*CPU time (secs)*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
06	2	10.81	17.16
06	2	12.80	21.38
07	2	17.75	29.70
07	2	14.05	23.48
08	2	21.60	57.68
08	2	33.16	47.88
11	2	45.21	72.28
12	2	161.21	167.56
13	2	148.68	185.55
15	2	220.31	337.90
18	2	477.38	691.35
20	2	724.86	1026.92
22	3	924.41	1398.88
24	3	705.28	1136.70
24	3	1057.95	1250.73
27	5	1230.25	1426.73
29	5	1638.57	1716.00
29	5	909.13	1591.93

Table E.5: CPU times (seconds - type A problems)

Type B			
Number of Jobs	*Machines per Agg. Res.*	*CPU time (secs)*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
06	2	10.03	17.68
06	2	9.53	17.22
07	2	10.43	20.00
08	2	22.62	35.20
08	2	16.38	28.85
09	2	24.72	42.02
12	2	59.67	112.48
13	2	105.55	175.55
15	2	233.48	283.20
15	2	182.90	322.45
18	2	333.93	344.98
20	2	371.00	470.50
20	3	262.80	371.95
23	3	780.05	1254.70
23	3	928.22	1181.47
27	5	689.67	1026.05
29	5	1443.48	1533.80
29	5	901.68	1354.10

Table E.6: CPU times (seconds - type B problems)

Type C			
Number of Jobs	*Machines per Agg. Res.*	*CPU time (secs)*	
		EXPLICIT-OAS-OUT	EXPLICIT-OAS-IN
06	2	8.38	15.37
07	2	9.63	30.83
07	2	10.85	17.78
09	2	22.50	48.07
09	2	21.75	45.37
09	2	22.98	43.60
10	2	28.33	57.12
11	2	38.00	54.27
12	2	49.08	144.08
15	2	115.05	270.35
17	2	157.72	309.70
17	2	256.10	491.72
20	3	156.60	376.90
21	3	397.62	969.62
22	3	631.48	922.58
26	5	385.10	540.32
27	5	1088.17	1317.82
29	5	381.05	780.55

Table E.7: CPU times (seconds - type C problems)

Appendix F

Regression Analysis

The regression analysis performed in order to investigate the behaviour of the versions EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN as the size of the problems increase is described in this appendix. This appendix complements the discussion presented in chapter 9 in terms of the cpu time analysis. The results are not very conclusive, though.

In order to find the curves that best explain the behaviour of the two versions of EXPLICIT, EXPLICIT-OAS-IN and EXPLICIT-OAS-OUT, in terms of cpu time and considering the least-squares criterion, several curves were fitted for each version considering the following settings:

- Type of jobs - for each set of points corresponding to a different type of job a different curve was fitted.
- Number of cases considered - two situations were considered to analyse the effect of the different number of machines:
  - ALL - the 18 points per type of job were considered for the fit.
  - 12 points set - only the 12 points set corresponding to the 2 machines cases was taken into account.
- Model - three types of simple regression models were considered:
  - Linear (Lin) - $cpu = a + b\ jobs$
  - Exponential (Exp) - $cpu = a\ exp\ (b\ jobs) \leftrightarrow ln\ (cpu) = ln\ (a) + b\ jobs$
  - Power Law - (P) - $cpu = a\ jobs^b \leftrightarrow ln\ (cpu) = ln\ (a) + b\ ln\ (jobs)$
    where:
    cpu - cpu time (seconds)
    $jobs$ - number of jobs to be scheduled
    a, b - parameters

Tables F.1, F.2, F.3 and F.4 synthesize the results of the different curves fitted, considering the different settings listed above and respectively for EXPLICIT-OAS-OUT and for

EXPLICIT-OAS-IN. In order to assess the results, *the coefficient of determination* (R^2), the *standard error of estimate* (*SEest*) and the *F-ratio* from the analysis of variance with the corresponding *degrees of freedom* (*DF*) and the respective F probability are included.

From the analysis of the tables, one can conclude that the linear fit is clearly the worst fit with respect to the several measures considered. Furthermore, this model is inadequate since the parameter a is significantly negative in all cases, though R^2 is significantly different from zero in all the cases.

In what concerns the exponential and the power law model, both curves fit well, though the power curve performs slightly better[1], except in the case of type C jobs, 12 points set, where the exponential curve fits better.

Note the different rate of growth of the cpu time relatively to th number of jobs, depending on the type of jobs, which is reflected on the different parameter b of the different curves. With respect to both versions, EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN, and when considering all of the cases, the cpu time increases at the slowest rate in the case of the type C jobs. Type A and type B jobs exhibit a similar rate of growth of the cpu time relatively to the number of jobs. For instance, in the case of the power law fit, and considering all the cases, the exponents of the fitted curves are 3.17, 3.188 and 3, respectively for type A, B and C jobs, and considering the version EXPLICIT-OAS-OUT. The same order of relation is found when considering the exponential model. For version EXPLICIT-OAS-IN, and considering again the power law model, the exponents of the fitted curves are 2.996, 2.961 and 2,849, respectively for type A, B and C jobs. EXPLICIT-OAS-OUT exhibits a greater rate of growth of the cpu time relatively to the number of jobs than EXPLICIT-OAS-IN.

Comparing the fits that consider all of the 18 cases per type of job with the ones that only consider the 12 cases per type of job, R^2 tends to be more significant when considering only the 12 observations. The explanation of this phenomenon is simple. The regression models adopted only consider *jobs* as the explicative variable for the variation of *cpu* (simple regression), without taking into consideration the number of machines. In the regressions performed on the 12 points sets, all the cases involve 2 machines per type of machine while in the regressions performed on all the cases, some cases involve 2 machines per type of aggregate resource, others 3 and others 5. The simple regression model does not consider the variation of number of machines as a source of explanation of the variation of *cpu*.

Considering the limitations of the models examined so far, due the fact that only a subset of the data was considered for the fits and also due to the fact that the variation of the number of machines was not taken into account for the explanation of the variation of the cpu time, a more global model is considered in the next paragraphs.

[1]The criterion adopted was the test of significance for prediction of *cpu*

Type	*Model*	*Cases*	*Curve*
A	Lin	All	$cpu = 57.94jobs - 456.544$
A	Lin	12	$cpu = 43.65jobs - 319.215$
A	Exp	All	$ln(cpu) = 0.2109jobs + 1.69138$
A	Exp	12	ln(cpu) = 0.3059 jobs jobs + 0.749647
A	P	All	$ln(cpu) = 3.17ln(jobs) - 3.26269$
A	P	12	$ln(cpu) = 3.46ln(jobs) - 3.89161$
B	Lin	All	$cpu = 47.76jobs - 409.416$
B	Lin	12	$cpu = 26.62jobs - 188.85$
B	Exp	All	$ln(cpu) = 0.2129jobs + 1.41886$
B	Exp	12	$ln(cpu) = 0.2925jobs + 0.59088$
B	P	All	$ln(cpu) = 3.188ln(jobs) - 3.59407$
B	P	12	$ln(cpu) = 3.323ln(jobs) - 3.88898$
C	Lin	All	$cpu = 30.81jobs - 259.022$
C	Lin	12	$cpu = 18.27jobs - 134.663$
C	Exp	All	$ln(cpu) = 0.1983jobs + 1.35447$
C	Exp	12	$ln(cpu) = 0.2898jobs + 0.411759$
C	P	All	$ln(cpu) = 3ln(jobs) - 3.44215$
C	P	12	$ln(cpu) = 3.164ln(jobs) - 3.81272$

Table F.1: Regression equations EXPLICIT-OAS-OUT

Type	*Model*	*Cases*	*DF*	R^2	*SEest*	*F-ratio*	*prob(F)*
A	Lin	All	(1,16)	0.8883	176.7163	127.2613	0.0000
A	Lin	12	(1,10)	0.8578	89.0161	60.3401	0.0000
A	Exp	All	(1,16)	0.9158	0.5501	173.9210	0.0000
A	Exp	12	(1,10)	0.9627	0.3016	258.1309	0.0000
A	P	All	(1,16)	0.9750	0.2998	623.3379	0.0000
A	P	12	(1,10)	0.9707	0.2673	331.3483	0.0000
B	Lin	All	(1,16)	0.8199	183.7160	72.8620	0.0000
B	Lin	12	(1,10)	0.9200	39.4710	115.0056	0.0000
B	Exp	All	(1,16)	0.9247	0.4985	196.5229	0.0000
B	Exp	12	(1,10)	0.9654	0.2786	278.7904	0.0000
B	P	All	(1,16)	0.9772	0.2743	685.8888	0.0000
B	P	12	(1,10)	0.9782	0.2209	449.4684	0.0000
C	Lin	All	(1,16)	0.6525	172.8162	30.0459	0.0001
C	Lin	12	(1,10)	0.8191	34.1483	45.2797	0.0001
C	Exp	All	(1,16)	0.9007	0.5058	145.2040	0.0000
C	Exp	12	(1,10)	0.9864	0.1352	727.6362	0.0000
C	P	All	(1,16)	0.9513	0.3544	312.3309	0.0000
C	P	12	(1,10)	0.9771	0.1755	427.5287	0.0000

Table F.2: Significance tests for prediction of the cpu time (EXPLICIT-OAS-OUT)

Type	*Model*	*Cases*	*Curve*
A	Lin	All	$cpu = 75.43jobsjobs - 576.25$
A	Lin	12	$cpu = 61.67jobs - 449.949$
A	Exp	All	$ln(cpu) = 0.1996jobs + 2.26142$
A	Exp	12	$ln(cpu) = 0.2872jobs + 1.38906$
A	P	All	$ln(cpu) = 2.996ln(jobs) - 2.41454$
A	P	12	$ln(cpu) = 3.236ln(jobs) - 2.93883$
B	Lin	All	$cpu = 62.17jobs - 517.453$
B	Lin	12	$cpu = 32.52jobs - 215.462$
B	Exp	All	$ln(cpu) = 0.1979jobs + 2.0727$
B	Exp	12	$ln(cpu) = 0.262jobs + 1.39938$
B	P	All	$ln(cpu) = 2.961ln(jobs) - 2.58073$
B	P	12	$ln(cpu) = 3ln(jobs) - 2.66782$
C	Lin	All	$cpu = 47.33jobs - 362.934$
C	Lin	12	$cpu = 36.74jobs - 267.564$
C	Exp	All	$ln(cpu) = 0.1864jobs + 2.20863$
C	Exp	12	$ln(cpu) = 0.2899jobs + 1.13751$
C	P	All	$ln(cpu) = 2.849ln(jobs) + -2.37567$
C	P	12	$ln(cpu) = 3.17ln(jobs) + -3.09912$

Table F.3: Regression equations EXPLICIT-OAS-IN

Type	*Model*	*Cases*	*DF*	R^2	*SEest*	*F-ratio*	*prob(F)*
A	Lin	All	(1,16)	0.9511	147.1311	311.1169	0.0000
A	Lin	12	(1,10)	0.8487	130.4041	56.1100	0.0000
A	Exp	All	(1,16)	0.9243	0.4913	195.3978	0.0000
A	Exp	12	(1,10)	0.9754	0.2285	396.4399	0.0000
A	P	All	(1,16)	0.9807	0.2483	811.3361	0.0000
A	P	12	(1,10)	0.9756	0.2277	399.4662	0.0000
B	Lin	All	(1,16)	0.8648	201.8042	102.3264	0.0000
B	Lin	12	(1,10)	0.9506	37.2847	192.4436	0.0000
B	Exp	All	(1,16)	0.9284	0.4509	207.5370	0.0000
B	Exp	12	(1,10)	0.9519	0.2962	197.8313	0.0000
B	P	All	(1,16)	0.9796	0.2410	766.8806	0.0000
B	P	12	(1,10)	0.9797	0.1923	483.0703	0.0000
C	Lin	All	(1,16)	0.7797	193.3560	56.6329	0.0000
C	Lin	12	(1,10)	0.8512	61.0975	57.2130	0.0002
C	Exp	All	(1,16)	0.8767	0.5372	113.7509	0.0000
C	Exp	12	(1,10)	0.9589	0.2386	233.5010	0.0000
C	P	All	(1,16)	0.9451	0.3584	275.5036	0.0000
C	P	12	(1,10)	0.9529	0.2556	202.3295	0.0000

Table F.4: Significance tests for prediction of the cpu time (EXPLICIT-OAS-IN)

In order to globally analyse the performance of the two versions in terms of cpu time, 4 models were tested considering the 54 cases:

- Simple regression (SR)- the models are analogous to the ones discussed above. The only difference is that the 54 cases are considered for the curve fitting. The linear regression was excluded. The number of jobs is the only explicative variable.
  - Exponential (Exp) - $cpu = a\ exp\ (b\ jobs) \leftrightarrow ln\ (cpu) = ln\ (a) + b\ jobs$
  - Power Law - (P) - $cpu = a\ x^b \leftrightarrow ln\ (cpu) = ln\ (a) + b\ ln\ (jobs)$
    where:
    cpu - cpu time (seconds)
    jobs - number of jobs to be scheduled
    a, b - parameters

  Figures F.1 and F.2 display the scatter diagrams and the fitted curves, exponential and power law curves, respectively for EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN.

- Multiple regression (MR) - in order to include the contribution of the variables number of machines and type of jobs for the explanation of the variation of the cpu time, a multiple regression analysis was performed. The multiple regression models have the following explicative variables:
  - *jobs* - number of jobs (as before)
  - *machines* - number of machines
  - *patA* - dummy variable. If jobs follow type A pattern, the variable has value 1, otherwise it has value 0.
  - *patB* - dummy variable. If jobs follow type B pattern, the variable has value 1, otherwise it has value 0 [2].

  The 2 multiple regression models are:
  - Exponential (Exp) - $cpu = aexp\ (bjobs) exp\ (cmachines) \rightarrow ln\ (cpu) = ln\ (a) + bJobs + cMachines + patA + patB$ [3]
  - Power Law - (P) - $cpu = ajobs^b machines^c \rightarrow ln\,(cpu) = ln\,(a) + bln\,(jobs) + cln\ (machines) + patA + patB$ [4]

Simple Regression Models
EXPLICIT-OAS-OUT

[2]To avoid multicolinearity, no dummy variable was considered for type C jobs. The jobs following the type C pattern are obtained by difference.

[3]The dummy variables were introduced after the logarithmic transformation of the original model

[4]The dummy variables were introduced after the logarithmic transformation of the original model

• Exponential Model

```
Analysis for 54 points of 2 variables:
Variable      ln(cpu)       jobs
Min            2.1263     6.0000
Max            7.4016    29.0000
Sum          256.3010   848.0000
Mean           4.7463    15.7037
SD             1.7142     7.7864

Correlation Matrix:
ln(cpu)        1.0000
jobs           0.9487     1.0000
Variable      ln(cpu)       jobs
```

The regression equation is:

$$ln\ (cpu) = 0.2089 jobs + 1.46652$$

The significance test for prediction:

Mult-R	R-Squared	SEest	F(1,52)	prob (F)
0.9487	0.9000	0.5473	467.9716	0.0000

• Power Law Model

```
Analysis for 54 points of 2 variables:
Variable      ln(cpu)  ln(jobs)
Min            2.1263     1.7918
Max            7.4016     3.3673
Sum          256.3010   141.5761
Mean           4.7463     2.6218
SD             1.7142     0.5325

Correlation Matrix:
ln(cpu)        1.0000
log(jobs)      0.9731     1.0000
Variable      ln(cpu)    ln(jobs)
```

The regression equation is:

$$ln\ (cpu) = 3.133 ln\ (jobs) - 3.46743$$

The significance test for prediction:

Mult-R	R-Squared	SEest	F(1,52)	prob (F)
0.9731	0.9470	0.3984	929.2981	0.0000

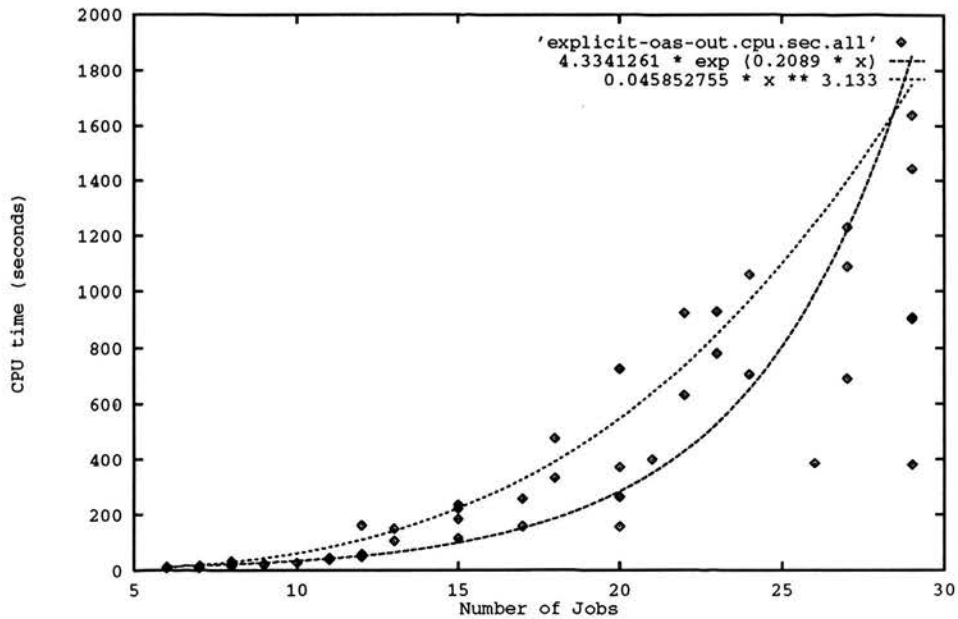

Figure F.1: CPU time (seconds) vs. Number of Jobs, simple regression models EXPLICIT-OAS-OUT

EXPLICIT-OAS-IN

- Exponential Model

Analysis for 54 points of 2 variables:

Variable	ln(cpu)	jobs
Min	2.7322	6.0000
Max	7.4478	29.0000

```
Sum            282.9016    848.0000
Mean             5.2389     15.7037
SD               1.5976      7.7864

Correlation Matrix:
ln(cpu)          1.0000
jobs             0.9526      1.0000
Variable       ln(cpu)        jobs
```

The regression equation is:

$$ln\ (cpu) = 0.1954 jobs + 2.16964$$

The significance test for prediction:

```
   Mult-R  R-Squared      SEest     F(1,52)   prob (F)
   0.9526     0.9074     0.4908    509.6297     0.0000
```

• Power Law Model

```
Analysis for 54 points of 2 variables:
Variable        ln(cpu)  log(jobs)
Min              2.7322     1.7918
Max              7.4478     3.3673
Sum            282.9016   141.5761
Mean             5.2389     2.6218
SD               1.5976     0.5325

Correlation Matrix:
ln(cpu)          1.0000
ln(jobs)         0.9807     1.0000
Variable       ln(cpu)    ln(jobs)
```

The regression equation is:

$$ln\ (cpu) = 2.942 ln\ (jobs) - 2.47538$$

The significance test for prediction:

```
   Mult-R  R-Squared      SEest     F(1,52)   prob (F)
   0.9807     0.9617     0.3155   1306.5457     0.0000
```

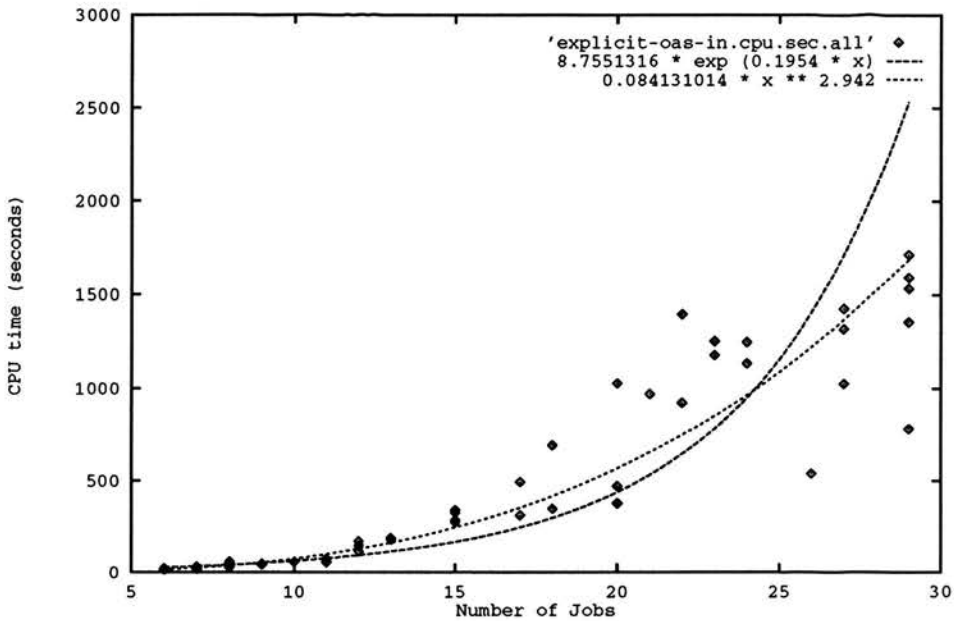

Figure F.2: CPU time (seconds) vs. Number of Jobs, simple regression models EXPLICIT-OAS-IN

Multiple Regression Models

EXPLICIT-OAS-OUT

- Exponential Model

```
Analysis for 54 points of 5 variables:
Variable      ln(cpu)       patA       patB    machines       jobs
Min            2.1263     0.0000     0.0000      2.0000     6.0000
Max            7.4016     1.0000     1.0000      5.0000    29.0000
Sum          256.3010    18.0000    18.0000    144.0000   848.0000
Mean           4.7463     0.3333     0.3333      2.6667    15.7037
SD             1.7142     0.4758     0.4758      1.1159     7.7864

Correlation Matrix:
ln(cpu)        1.0000
patA           0.1230     1.0000
patB           0.0326    -0.5000     1.0000
machines       0.6760     0.0000     0.0000      1.0000
jobs           0.9487     0.0170     0.0272      0.8462     1.0000
Variable      ln(cpu)       patA       patB    machines       jobs
```

The regression equation is:

$$ln\ (cpu) = 0.4765patA + 0.2274patB - 0.6673machines + 0.2889jobs + 1.75427$$

The significance test for prediction:

Mult-R	R-Squared	SEest	F(4,49)	prob (F)
0.9847	0.9696	0.3107	390.9861	0.0000

• Power Law Model

Analysis for 54 points of 5 variables:

Variable	ln(cpu)	patA	patB	ln(machines)	ln(jobs)
Min	2.1263	0.0000	0.0000	0.6931	1.7918
Max	7.4016	1.0000	1.0000	1.6094	3.3673
Sum	256.3010	18.0000	18.0000	49.3257	141.5761
Mean	4.7463	0.3333	0.3333	0.9134	2.6218
SD	1.7142	0.4758	0.4758	0.3479	0.5325

Correlation Matrix:

ln(cpu)	1.0000				
patA	0.1230	1.0000			
patB	0.0326	-0.5000	1.0000		
ln(machines)	0.7071	-0.0000	-0.0000	1.0000	
ln(jobs)	0.9731	-0.0029	0.0255	0.7714	1.0000
Variable	ln(cpu)	patA	patB	ln(machines)	ln(jobs)

The regression equation is:

$$ln\ (cpu) = 0.6191patA + 0.3304patB - 0.5132ln\ (machines) + 3.386ln\ (jobs) - 3.97774$$

The significance test for prediction:

Mult-R	R-Squared	SEest	F(4,49)	prob (F)
0.9869	0.9739	0.2882	456.6502	0.0000

EXPLICIT-OAS-IN

• Exponential Model

Analysis for 54 points of 5 variables:

Variable	ln(cpu)	patA	patB	machines	jobs
Min	2.7322	0.0000	0.0000	2.0000	6.0000
Max	7.4478	1.0000	1.0000	5.0000	29.0000
Sum	282.9016	18.0000	18.0000	144.0000	848.0000
Mean	5.2389	0.3333	0.3333	2.6667	15.7037
SD	1.5976	0.4758	0.4758	1.1159	7.7864

Correlation Matrix:

log(cpu)	1.0000				
patA	0.0867	1.0000			
patB	-0.0002	-0.5000	1.0000		
machines	0.6739	0.0000	0.0000	1.0000	
jobs	0.9526	0.0170	0.0272	0.8462	1.0000
Variable	ln(cpu)	patA	patB	machines	jobs

The regression equation is:

$$ln\ (cpu) = 0.2038patA - 0.02119patB - 0.6625machines + 0.2756jobs + 2.61658$$

The significance test for prediction:

Mult-R	R-Squared	SEest	F(4,49)	prob (F)
0.9864	0.9730	0.2729	441.8119	0.0000

• Power Law Model

Analysis for 54 points of 5 variables:

Variable	ln(cpu)	patA	patB	ln(machines)	ln(jobs)
Min	2.7322	0.0000	0.0000	0.6931	1.7918
Max	7.4478	1.0000	1.0000	1.6094	3.3673
Sum	282.9016	18.0000	18.0000	49.3257	141.5761
Mean	5.2389	0.3333	0.3333	0.9134	2.6218
SD	1.5976	0.4758	0.4758	0.3479	0.5325

Correlation Matrix:

ln(cpu)	1.0000				
patA	0.0867	1.0000			
patB	-0.0002	-0.5000	1.0000		
ln(machines)	0.7078	-0.0000	-0.0000	1.0000	
ln(jobs)	0.9807	-0.0029	0.0255	0.7714	1.0000
Variable	ln(cpu)	patA	patB	ln(machines)	ln(jobs)

The regression equation is:

$$ln\ (cpu) = 0.34patA + 0.07746patB - 0.5505ln\ (machines) + 3.219ln\ (jobs) - 2.83684$$

The significance test for prediction:

Mult-R	R-Squared	SEest	F(4,49)	prob (F)
0.9880	0.9760	0.2571	499.1790	0.0000

EXPLICIT-OAS-OUT

- Simple Regression, Exponential Model (SR-Exp)

$$ln\ (cpu) = 0.2089jobs + 1.46652$$

- Simple Regression, Power Law Model(SR-P)

$$ln\ (cpu) = 3.133ln\ (jobs) - 3.46743$$

- Multiple Regression, Exponential Model (MR-Exp)

$$ln\ (cpu) = 0.4765patA + 0.2274patB + -0.6673machines + 0.2889jobs + 1.75427$$

- Multiple Regression, Power Law Model (MR-P)

$$ln\ (cpu) = 0.6191patA + 0.3304patB - 0.5132ln\ (machines) + 3.386ln\ (jobs) - 3.97774$$

EXPLICIT-OAS-IN

- Simple Regression, Exponential Model(SR-Exp)

$$ln\ (cpu) = 0.1954 jobs + 2.16964$$

- Simple Regression, Power Law Model(SR-P)

$$ln\ (cpu) = 2.942 ln\ (jobs) - 2.47538$$

- Multiple Regression, Exponential Model (MR-Exp)

$$ln\ (cpu) = 0.2038 patA - 0.02119 patB - 0.6625 machines + 0.2756 jobs + 2.61658$$

- Multiple Regression, Power Law Model (MR-P)

$$ln\ (cpu) = 0.34 patA + 0.07746 patB - 0.5505 ln\ (machines) + 3.219 ln\ (jobs) - 2.83684$$

EXPLICIT-OAS-OUT

Model	Mult-R	R-Squared	SEest	F	DF prob (F)
SR-Exp	0.9487	0.9000	0.5473	467.9716	0.0000
SR-P	0.9731	0.9470	0.3984	929.2981	0.0000
MR-Exp	0.9847	0.9696	0.3107	390.9861	0.0000
MR-P	0.9869	0.9739	0.2882	456.6502	0.0000

EXPLICIT-OAS-IN

Model	Mult-R	R-Squared	SEest	F	DF prob (F)
SR-Exp	0.9526	0.9074	0.4908	509.6297	0.0000
SR-P	0.9807	0.9617	0.3155	1306.5457	0.0000
MR-Exp	0.9864	0.9730	0.2729	441.8119	0.0000
MR-P	0.9880	0.9760	0.2571	499.1790	0.0000

The regression equations of the 4 models adopted and the corresponding significance tests for prediction are listed above, respectively for EXPLICIT-OAS-OUT and EXPLICIT-OAS-IN. From the analysis of these results, the multiple regression models perform slightly better than the simple regression models. The values of R^2 associated with the multiple regression models are greater than the values of R^2 associated with the simple regression models and the values of $SEest$ associated with the multiple regression models are smaller than the corresponding values produced by the simple regression models.

Assuming the multiple regression models as representing better fits than the simple regression models, the question is which model, exponential vs. power law, better fits the data? In other words, what is the relation between the cpu time and the explicative variables. What is the behaviour of the cpu time as the size of the problems increase. Is it exponential? Is it polynomial? From the results of the fits, considering the 54 cases, the best fit seems to be the power law curve. This is also true for the other models tested, excepted in one case, type C jobs, for the case where the fit was performed considering the 12 points set. However, it seems important to mention that, though the least squares is a convenient and easy method to the problem of curve fitting, in several aspects it is an arbitrary method and it is based on an *empirical* relationship, i.e it depends on the particular data that is considered for the curve fitting. A more theoretical study of the complexity of the approach described in this thesis seems to be important. Especially for the version EXPLICIT-OAS-IN, due to the inclusion of the Operational Agents to which an optimal algorithm for minimising the maximum lateness was assigned, there are strong reasons to believe that the behaviour of the system is exponential with the size of the problems. Nevertheless, even exponential algorithms may perform sufficiently well when the size of the problems is small. An important principle underlying the approach "achieving global coherence by exploiting conflict" is "divide to conquer", i.e solve a large problem by splitting it into several smaller, more manageable and easier to understand problems.

Bibliography

[Adams *et al* 88] J. Adams, E. Balas, and D. Zawack. The Shifting Bottleneck Procedure For Job Shop Scheduling. *Management Science*, 34(3):391–401, 1988.

[Allen *et al* 90] James Allen, James Hendler, and Austin Tate. *Readings in Planning.* Morgan Kaufmann, 1990.

[Arabeyre *et al* 69] J. P. Arabeyre, J. Fearnley, F. C. Steiger, and W. Teather. The Airline Crew Scheduling Problem: A Survey. *Transportation Science*, 3:141–163, 1969.

[Baker 74] K. R. Baker. *Introdution to Sequencing and Scheduling.* Wiley, 1974.

[Balas & Saltzman 91] E. Balas and M. Saltzman. An algorithm for the three-index assignment problem. *Operations Research*, 39(3):150–161, 1991.

[Ball & Roberts 85] M. Ball and A. Roberts. A Graph partitioning Approach to Airline Crew Scheduling. *Transportation Science*, 19:107–126, 1985.

[Ball *et al* 81] M. Ball, L. Bodin, and R. Dial. A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles. *Transportation Science*, 17:4–31, 1981.

[Bensana *et al* 88] E. Bensana, G. Bel, and D. Dubois. Opal: A multi-knowledge-based system for industrial job-shop sacheduling. *International Journal of Production Research*, 26(5):795–819, 1988.

[Berry 91] Pauline M. Berry. A Predictive Model for Satisfying Conflicting Objectives in Scheduling Problems. PhD Thesis, University of Strathclyde, Glasgow, Scotland, 1991.

[Berry 92] Pauline M. Berry. A Predictive Model for Satisfying Conflicting Objectives in Scheduling Problems. *AI Magazine*, 3(1), Spring 1992.

[Blackstone *et al* 82] J. H. Blackstone, D. T. Phillips, and G. L. Hogg. Heuristics in Job Shop Scheduling. *International Journal of Production Research*, 20(1):27–45, 1982.

[Blazewicz88 *et al* 88] J. Blazewicz88, G. Finke, R. Haupt, and G. Schmidt. New trends in machine scheduling. *European Journal of Operational Research*, 37(3):303–317, 1988.

[Blazwicz *et al* 91] J. Blazwicz, M. Dror, and J. Weglarz. Mathematical programming formulations for machine scheduling - a survey. *European Journal of Operational Research*, 51(3):283–300, 1991.

[Bodin *et al* 83] L. Bodin, A Assad, and M. Ball. Special Issue - Routing And Scheduling of Vehicles and Crews - The State of The Art. *Computers and Operations Research*, 10(2):1, 1983.

[Bond & Les Gasser 88] Alan H. Bond and Les Gasser. An Analysis of Problems and Research in DAI. In Alan H. Bond and Les Gasser, editors, *Readings in Artificial Intelligence*. Morgan Kaufmann, 1988.

[Buchanan *et al* 88] J. T. Buchanan, Peter Burke, John Costello, and Patrick Prosser. A Distributed Asynchronous Scheduler. Technical Report AISL-32-88, University of Strathclyde, 1988.

[Buchanan *et al* 89] J. T. Buchanan, Peter Burke, John Costello, and Patrick Prosser. A Distributed Asynchronous Hierarchical Problem Solving Architecture Applied to Plant Scheduling. In *Conference on AAIE*, Cambridge, U.K., 1989.

[Burkard & Derigs 80] R. Burkard and V. Derigs. Assignment and Matching Problems: Solution Methods with Fortran Programs. In *Lecture-notes in Economics and Mathematical Systems*. Springer Verlag, 1980.

[Burke & Prosser 89] Peter Burke and Patrick Prosser. A Distributed Asynchronous System for Predictive and Reactive Scheduling. Technical Report AISL-42-89, University of Strathclyde, October 1989.

[Burke 89] Peter Burke. *Scheduling in Dynamic Environments*. Unpublished PhD thesis, University of Strathclyde, 1989.

[Buxey 89] G. Buxey. Production scheduling - practice and theory. *European Journal of Operational Research*, 39(1):17–31, 1989.

[Carlier 82] J. Carlier. The One-Machine Sequencing Problem. *European Journal of Operations Research*, 11:42–47, 1982.

[Carraresi & Gallo 84] P. Carraresi and G. Gallo. Network Models For Vehicle and Crew Scheduling. *European Journal of Operational Research*, 16(2):139–151, 1984.

[Charalambous & Hindi 91] O. Charalambous and K. Hindi. A Review Of Artificial Intelligence Based Job Shop Scheduling Systems. *Information and Decision Technologies*, 17(3):189–202, 1991.

[Charniak & McDermott 85] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley Publishing Company, 1985.

[Cheng & Sin 90] T. Cheng and C. Sin. A state of the art review of parallel machine sequencing reserach. *European Journal of Operational Research*, 47(3):271–292, 1990.

[Christofides 75] N. Christofides. *Graph Theory: An Algorithmic Approach.* Academic Press, 1975.

[Collinot *et al* 88] Anne Collinot, Claude Le Pape, and Gerard Pinoteau. SONIA: A Knowledge-Based Scheduling System. *Artificial Intelligence in Engineering*, 3(2):86–94, April 1988.

[Cook 71] S. A. Cook. The Complexity of Theorem Proving Procedures. In *Proceedings of The Third Annual ACM Symposium on The Theory of Computing*, pages 151–158, 1971.

[DARPA 90] DARPA. *Proceedings of Workshop on Innovative Approaches to Planning, Scheduling and Control.* Morgan Kaufmann, 1990.

[Davis & Patterson 75] E. W. Davis and J. H. Patterson. A Comparison of Heuristics and Optimum Solutions in Resource Constrained Project Scheduling. *Management Science*, 21(8):944–955, 1975.

[Davis & Smith 83] R. Davis and R. G. Smith. Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, 29:63–109, 1983.

[Davis 87] E. Davis. Constarint Propagation with Interval Labels. *Artificial Intelligence*, 32:281–331, 1987.

[de Kleer 86] Johan de Kleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[Decker *et al* 89] Keith S. Decker, Edmund H. Durfee, and Victor R. Lesser. Evaluating Research in Cooperative Distributed Problem Solving. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, pages 485–519. Morgan Kaufmann, 1989.

[Dekel & Sahni 83] E. Dekel and S. Sahni. Parallel Scheduling Algorithms. *Operations Research*, 31(1):24–49, 1983.

[Desimone *et al* 90] Roberto Desimone, Brian Drabble, and Howard Beck. Planning and Scheduling Course. Course Notes, AIAI, University of Edinburgh, 1990.

[Detcher 86] Rina Detcher. Learning While Searching in Constraint Satisfaction Problems. In *Proceedings of AAAI*, pages 178–183, 1986.

[Dileepan & Sen 88] P. Dileepan and T. Sen. Bicreterion Static Scheduling Research for a Single Machine. *OMEGA*, 16(1):53–59, 1988.

[Doyle 81] J. Doyle. A Truth Maintenance Systems. In N. J. Nilsson, editor, *Readings in Artificial Intelligence.* Morgan Kaufmann, 1981.

[Drummond & Tate 89] Mark Drummond and Austin Tate. Ai planning: A tutorial and review. Technical Report AIAI-TR-30, Artificial Intelligence Applications Institute (AIAI), January 1989.

[Durfee 89] Edmund H. Durfee. *Coordination of Distributed Problem Solvers.* Klumer Academic Publishers, 1989.

[Durfee *et al* 87] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Cooperation through Communication in a Distributed Problem Solving Network. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, pages 29–58. Morgan Kaufmann, 1987.

[Eglese 86] R. W. Eglese. Heuristics in Operational Research. *Recent Developments in Operational Research*, pages 49–67, 1986.

[Fendley 68] L. G. Fendley. Toward the Development of a Complete Multiproject Scheduling System. *J. Indust. Engineering*, 19(10):505–515, 1968.

[Fox & Smith 84] Mark S. Fox and Stephen F. Smith. ISIS - A Knowledge-Based System for Factory Scheduling. *Expert Systems*, 1(1):25–48, July 1984.

[Fox & Sycara 90] Mark S. Fox and Katia Sycara. The CORTES Project: A Unified Framework for Planning, Scheduling and Control. In *Proceedings of Workshop on Innovative Approaches to Planning, Scheduling and Control*, 1990.

[Fox 88] Mark S. Fox. An Organizational View of Distributed Systems. In Alan H. Bond and Les Gasser, editors, *Readings in Artificial Intelligence.* Morgan Kaufmann, 1988.

[French 82] S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* Ellis Horwood, Chichester, England, 1982.

[Garey & Johnson 79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., San Francisco, 1979.

[Gaschnig 77] J. Gaschnig. A General Backtracking Algorithm That Eliminates Most Redundant Tests. In *Proceedings of IJCAI*, 1977.

[Gerbracht 78] R. Gerbracht. A New Algorithm for Very Large Crew Pairing Problems. In *Proceedings of AGIFORS Group Meeting*, Continental Airlines, 1978.

[Gerbracht 85] R. Gerbracht. A Monthly Pairing Optimisation. In *Proceedings of AGIFORS Group Meeting*, Airline Information Systems, 1985.

[Gere 66] W. S. Gere. Heuristics in Job Shop Scheduling. *Management Science*, 13(3):167–190, 1966.

[Gomes & Almeida 88] Carla Pedro Gomes and M. T. Almeida. Pairing Generation - A Graph Partitioning Approach to a Short Haul Fleet Problem. In *Proceedings of AGIFORS Group Meeting*, Copenhagen, 1988.

[Gomes & Beck 92] Carla Pedro Gomes and Howard Beck. Synchronous and Asynchronous Factory Scheduling. To appear in Information Technology - Journal of the Singapore Computer Society, 1992.

[Gomes 87] Carla Pedro Gomes. Escalonamento de Tripulacoes - Geracao de Cadeias de Voo para uma Frota de Medio Curso. Master Thesis, Universidade de Lisboa - Instituto Superior de Economia, 1987.

[Gondran & Minoux 84] M. Gondran and M. Minoux. *Graphs and Algorithms.* John Wiley & Sons, 1984.

[Graves 81] S. Graves. A Review of production Scheduling. *Operations Research*, 29(4):646–675, 1981.

[Gupta & Kyparisis 87] S. Gupta and J. Kyparisis. Single Machine Scheduling Research. *Omega - International Journal of Management Science*, 15(3):207–227, 1987.

[Haralick & Elliot 80] Robert M. Haralick and Gordon L. Elliot. Increasing Tree Search Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14:263–314, October 1980.

[Hastings *et al* 82] N. A. J. Hastings, P. H. Marshall, and R. J. Willis. Schedule Based MRP: An Integrated Approach to Production Scheduling and Materials Requirement Planning. *Journal of The Operational Research Society*, 33:1021–1029, 1982.

[Hax & Meal 75] A. C. Hax and H. C. Meal. Hierarchical integration of production planning and scheduling. *Studies in management Science*, 1:53–69, 1975.

[Hayes-Roth 85] Barbara Hayes-Roth. A Blackboard Architecture for Control. *Artificial Intelligence Journal*, 3:251–321, 1985.

[Hern 88] Luis Eduardo Castillo Hern. On Distributed Artificial Intelligence. *Knowledge Engineering Review*, 3(1):21–57, March 1988.

[Jacobs 84] F. R. Jacobs. OPT Uncovered: Many Production Planning and Scheduling Concepts Can Be Applied With or Without Software. *Industrial Engineering*, 16(10):32–41, 1984.

[Kan 76] Rinnooy Kan. *Machine Scheduling Problems: Classification, Complexity and Computations.* Nijhof, The Hague, 1976.

[Karp 72] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 95–103. Plenum Press, New York, 1972.

[Kornfeld & Hewitt 88] W. A. Kornfeld and C. E. Hewitt. The Scientific Community Metaphor. In Alan H. Bond and Les Gasser, editors, *Readings in Artificial Intelligence*, pages 311–321. Morgan Kaufmann, 1988.

[Kurtulus & Narula 85] I. Kurtulus and S. Narula. Multi Project Scheduling - Analysis of Project Performance. *IIE Transactions*, 17(1):58–66, 1985.

[Lawler *et al* 82] E. L. Lawler, J. K. Lenstra, and Rinnooy Kan. Recent Developments in Deterministic Sequencing Scheduling. In M. A. H. Demspster, J. K. Lenstra, and A. H. G. Rinnooy Kan, editors, *Deterministic and Stochastic Scheduling.* D. Reidel Publishing, Dordrecht, The Netherlands, 1982.

[Lawrence 87] Andrew Lawrence. MRP, OPT, JIT: The Facts. *Industrial Computing*, September 1987.

[Le Pape 85] C. Le Pape. SOJA: A Daily Workshop Scheduling System. In *Proceedings of the Fifth Technical Conference of the BCS Specialist Group on Expert Systems*, Warwick, Great Britain, 1985.

[Le Pape 91] Claude Le Pape. Architectural Issues in the Design of (Future) AI-Based Schedulers. In *Proceedings of the 10th SIG Planning*, Cambridge, 1991.

[Liu 88] Bing Liu. Scheduling via Reinforcement. *Artificial Intelligence in Engineering*, 3(2):78–85, April 1988.

[Lockyer 84] K. G. Lockyer. *Critical path analysis and other project network techniques.* London Pitman, 1984.

[Mackworth 77] Alan K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.

[Malone & Smith 84] Thomas W. Malone and Stephen F. Smith. Tradeoffs in Designing Organizations: Implications For New Forms Of Human Organizations and Computer Systems. Working Paper WP 1541-84, Sloan School of Management, MIT, 1984.

[Marsten & Shepardson 81] R. E. Marsten and F. Shepardson. Exact Solution of Scheduling Problems using Set Partitioning Model: Recent Successful Applications. *Networks*, 11:165–177, 1981.

[Marsten *et al* 79] R. E. Marsten, M. R. Muller, and C. C. Killion. Crew Planning at Flying Tiger: A Successful Application of Integer Programming. *Management Science*, 35(12):1175–1183, 1979.

[McKay *et al* 88] Kenneth N. McKay, Frank R. Safayeni, and John A. Buzacott. Job-Shop Scheduling Theory: What Is Relevant? *Interfaces*, 18(1):84–90, July-August 1988.

[McMahon & Florian 75] Graham McMahon and Michael Florian. On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness. *Operations Research*, 23:475–481, 1975.

[Minoux 84] M. Minoux. Column Geneartion Techniques in Combinatorial Optimisation. In *Proceedings of AGIFORS Group Meeting*, Air France, 1984.

[Minoux 85] M. Minoux. *Mathematical Programming: Theory and Algorithms.* John Wilwey & Sons, 1985.

[Nadel 86] B. A. Nadel. The general consistent labelling (or constraint satisfaction) problem. Technical Report DCS-TR-170, Laboratory for Computer Science, Rutgers University, New Brunswick, Laboratory for Computer Science, Rutgers University, New Brunswick, NJ 08903, 1986.

[Newell & Simon 76] A. Newell and H. A. Simon. Computers Sciences as Empirical Inquiry: Symbols and Search. *Communications of the ACM*, 19(3):113–126, 1976.

[Panwalkar & Iskander 77] S. S. Panwalkar and Wwafik Iskander. A Survey of Scheduling Rules. *Operations Research*, 25(1):45–61, 1977.

[Park *et al* 84] Y. B. Park, C. D. Pedgen, and E. E. Enscore. A Survey and Evaluation of Static Flow Shop Scheduling Heuristics. *International Journal of Production Research*, 22(1):127–141, 1984.

[Parunak 87] H. Van Dyke Parunak. Manufacturing Experience with the Contract Net. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, pages 285–310. Morgan Kaufmann, 1987.

[Polya 48] G. Polya. *How To Solve It.* Princeton University Press, Princeton, N. J., 1948.

[Prosser 88] Patrick Prosser. Reactive Factory Scheduling as a Dynamic Constraint Satisfaction Problem. Technical Report AISL-31-88, University of Strathclyde, August 1988.

[Roberts 86] R. Roberts. Crew Management Revisited. In *Proceedings of AGIFORS Group Meeting*, Sita, London, 1986.

[Sadeh 91] Norman Sadeh. *Look-ahead Techniques for Micro-oportunistic job shop scheduling.* Phd, School of Computer Science, Carnegie Mellon University, 1991. (CMU-CS-91-102).

[Schrage 71] L. Schrage. Obtaining Optimal Solutions to Resource Constrained Network Scheduling. Manuscript, 1971.

[Serafini *et al* 88] P. Serafini, W. Ukovich, H. Kirchner, F. Giardina, and F. Tiozzo. Job-shop Scheduling: a Case Study. In RF Archetti, M. Lucertini, and P. Serafini, editors, *Operations Research Models In FMS.* Springer, Vienna, 1988.

[Simon & Newell 58] H. A. Simon and A. Newell. Heuristic Problem Solving: The Next Advance In Operations Research. *Operations Research*, 7:1–10, 1958.

[Smith 87] Stephen F. Smith. A Constraint-Based Framework for Reactive Management of Factory Schedules. In *Proceedings of the International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, May 1987.

[Smith *et al* 86] Stephen F. Smith, Mark S. Fox, and Peng Si Ow. Constructing and Maintaining Detailed Production Plan: Investigations into the Developmemt of Knowledge-Based Factory Scheduling Systems. *AI Magazine*, 7(4):45–61, Fall 1986.

[Stallman & Sussman 77] R. M. Stallman and G. J. Sussman. Forward Reasoning and Dependency Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*, 9:138–196, 1977.

[Tate *et al* 90] Austin Tate, James Hendler, and Mark Drummond. A Review of AI Planning Techniques. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*. Morgan Kaufmann, 1990.

[Willis 85] R. J. Willis. Critical path-analysis and resource constrained project scheduling - theory and practice. *European Journal of Operational Research*, 21(2):149–155, 1985.

[Zob 79] A. P. Zob. International Airline Crew Scheduling. In *Proceedings of AGIFORS Group Meeting*, Boeing Company, 1979.